НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:

Завідувач кафедри

_____ С. Г. СТІРЕНКО

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних систем»

спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Веб додаток моделі загального публікаційного блогу»

Виконав (-ла):
студент IV курсу, групи ІП-64
Сєнін Юрій Ігорович _____

Керівник:
Асистент
Аленін Олег Ігорович _____

Консультант з нормо контролю:
Професор, доктор технічних наук
Сімоненко Валерій Павлович _____

Рецензент:

_____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМ. ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення комп'ютерних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ *Сергій СТІРЕНКО*
(підпис)
"___"_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту

Сєніну Юрію Ігоровичу

1. Тема проєкту «*Веб додаток моделі загального публікаційного блогу»*

керівник проєкту Аленін Олег Ігорович, аспірант, затверджені наказом по університету від *«07»* травня 2020р. №__1081-с__

2. Термін здачі студентом закінченого роботи__*27 травня 2020р.*
3. Вихідні дані до проєкту *): дивитись в технічне завдання.*

4. Зміст пояснювальної записки: *Дослідження і вивчення запропонованих рішень. Вибір технологій розробки. Проєктування СУБД. Реалізація запропонованого веб-додатка.*

5. Консультант роботи, з вказівкою розділів роботи, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормо контроль	Сімоненко		

6. Дата видачі завдання *01.09 .2019 року*

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту(роботи)	Примітки
1.	Затвердження теми роботи	10.12.2019-15.12.2019	
2.	Вивчення та аналіз завдання	15.12.2019-15.03.2020	
3.	Розробка архітектури та загальної структури системи	15.03.2020-25.03.2020	
4.	Розробка структур окремих Підсистем	25.03.2020-05.04.2020	
5.	Програмна реалізація системи	05.04.2020-15.04.2020	
6.	Оформлення пояснювальної	15.04.2020-20.05.2020	
7.	Захист програмного продукту	25.04.2020	
8.	Перед захист	28.05.2020	
9.	Захист	19.06.2020	

Студент Юрій СЄНІН _____

(підпис)

Керівник Олег АЛЕНІН_____

(підпис)

Анотація

У моїй дипломній роботі був створений додаток який дозволяє ефективно та зручно вести свій блог.

Даний веб-додаток був розроблений для зручного ведення блогу з можливістю давання відгуків про деякі статті та була реалізована можливість підписників певного блогера.

Програмний продукт дозволяє зручно і ефективно додавати нові статті, а також змінювати їх і доповнювати. У даній програмі присутні ролі як адміністратор(модераторів), так і звичайний користувач. При чому Адміністратор має можливість додавання нових модераторів, які в свою чергу будуть відслідковувати, редагувати і видаляти пости звичайних користувачів, які будуть надісланні.

Веб-додаток був створений у середовищі IntelliJ IDEA 2020.1 використовуючи мову програмування Java.

Annotation

In my diploma work, the application was created that allows you effectively and conveniently to run your blog.

This web application was developed for easy blogging with the ability to give a feedback on some articles and it was implemented as a subscriber to a particular blogger.

The software product allows you to conveniently and efficiently add new articles, as well as change them and add. In this web-application there are roles as an administrator (moderator) and a regular user. Moreover, the Admin has the ability to add new moderators, which will observe, edit and delete posts of regular users those will be sent.

Web application was created in the IntelliJ IDEA 2020.1 environment using the Java programming language.

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ІАЛЦ.467100.001 ВП	Відомість проєкту	1	
3	А4	ІАЛЦ.467100.002 ТЗ	Технічне завдання	4	
4	А4	ІАЛЦ.467100.003 ПЗ	Пояснювальна записка	71	
5	А3	ІАЛЦ.467100.004 Д1	Алгоритм дій програмного забезпечення	1	
6	А3	ІАЛЦ.467100.005 Д2	Функціональна схема(діаграма класів)	1	
7	А3	ІАЛЦ.467100.006 Д3	Структурна схема веб-застосунку	1	
8	А4	ІАЛЦ.467100.007 Д4	Текст програми	13	

Зм.	Арк.	№ документна	Підпис	Дата	ІАЛЦ.467100.001 ВП		
Розроб.		Сєнін Ю.І.			Веб-додаток моделі загального публікаційного блогу Відомість дипломного проєкту	Літ.	Аркуш 5 / Акрушів 1
Перевір.		Аленін О.І.					
Н. Контр.		СимоненкоВ.П.				НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ, гр.ІП-64	
Затверд.							

ТЕХНІЧНЕ ЗАВДАННЯ

до дипломної роботи

освітньо-кваліфікаційного рівня бакалавр

на тему: “Веб-додаток загального публікаційного блогу”

Київ – 2020 року

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється на розробку і в майбутньому подальшу підтримку запропонованого програмного продукту ”Веб додаток моделі загального публікаційного блогу ”.

Область застосування даного програмного забезпечення – може являтися в невеликих компаніях, а також групи людей з спільними інтересами.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки даного продукту є завдання для виконання роботи кваліфікаційно-освітнього рівня «бакалавр інженерії програмного забезпечення», який був затверджений факультетом “Інформатики та обчислювальної техніки” кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут ім. Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою і призначенням написання даного програмного продукту є ідея наштовхнути людей почати вести і обговорювати питання на тему, яка буде їм до вподоби.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки дипломного проєкту є обговорення різних блогів з програмування на мові Java та JavaScript, а також науково-технічна література. Статті по фреймворку Spring.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до продукту, що розробляється

Розроблене програмне забезпечення має виконувати такі вимоги:

- Надати необхідний функціонал для швидкого і ефективного користуванням сайтом, а також зручний і зрозумілий інтерфейс;
- Пошук статей по запропонованому тегу;
- Додавання нових та редагування старих статей;
- Можливість поставити уподобання до статті яка вам сподобалася;
- Можливість підписатися на користувача статті якого вам до вподоби.

5.2. Вимоги до програмного забезпечення

- ОС Windows або Mac
- Java 8+
- Intellij IDEA 2020.1
- Node-v8.17.0
- PostgreSQL v11

5.3. Вимоги до апаратної частини

- ЦП не менше ніж Intel(R) Core (TM) i3-3130M
- ROM не менше ніж 64 ГБ.
- RAM не менше ніж 2 ГБ.

ЕТАПИ РОЗРОБКИ

	Дата
Аналіз і вивчення літератури	25.11.2019
Складання технічного завдання	02.12.2019
Дослідження та вибір ресурсів для розробки	20.01.2020
Проєктування бази даних	24.01.2020
Розробка програмного забезпечення	01.02.2020
Тестування програмного забезпечення	27.04.2020
Налагодження та виправлення помилок	11.05.2020
Написання документації дипломного проєкту	06.06.2020

Пояснювальна записка
до дипломного проєкту
на тему: «Веб додаток моделі загального публікаційного блогу»

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

Front end – користувацька частина взаємодії з програмою.

Back end – серверна частина, для обробки бізнес логіки програми.

Java - об'єктно-орієнтована, крос платформна мова програмування, яка призначена для розробки програмного забезпечення.

Spring – один із самих популярних фреймворків мови програмування Java для легкого створення веб-додатків з додатковими можливостями.

Intellij Idea – середовище розробки для різних мов програмування частіше за все це Java.

PostgreSQL – об’єктно-реляційна система, яка використовується для керуванням даних в СУБД.

SQL - мова програмування, яка призначена для взаємодії бази даних з програмою.

JVM – Java Virtual Machine.

HTML –мова розмітки призначена для розробки веб-сторінок.

Фреймворк – це вже готовий програмний код написаний іншими розробниками, який полегшує створення.

HTTP – протокол який призначений для надсилання та отримання запитів.

HTML – стандартна мова розмітки для розробки веб-сторінок.

Spring Data – один із модулів фреймворку Spring, який на взаємодію даних

Spring Security – один із модулів фреймворку Spring, який відповідає за авторизацію та аутентифікацію

Spring Boot – один із модулів фреймворку Spring, який відповідає за взаємодію між компонентами та надає розширені можливості при управлінні та запуску програми.

MVC – шаблон проєктування Model View Controller, який розділяє програму на 3 окремі частини: model, view і controller.

ВСТУП

На сьогоднішній день сучасний світ влаштований так, що так чи інакше усі події контролюються великим потоком даних інформації. Ця інформація може бути різноманітного роду. Люди проводять надзвичайно багато часу в соціальних мережах, спілкуючись на різні теми. Однак не меншу кількість часу люди проводять переглядаючи різні блоги. Для прикладу, у сьогодення блоги мають значну перевагу над месенджерами у поширенні інформації в маси, адже людям набагато зручніше обговорювати різні теми залишаючи свої пости серед вже створеними. Це в свою чергу дає можливість великій кількості людей обговорювати, сперечатися або ж доносити свою ідею в маси. Відповідно до цього блоги є більш зручним середовищем для різноманітного роду обговорень, що можуть стосуватися будь-чого.

Ось представленні 5 основних переваг для ведення блогу:

- Нетворкінг. Бути блогером це означає, завжди спілкуватися зі своєю аудиторією, тож слід сказати, що спілкування в мережі сприяє новим та покращує старі знайомства. Якщо існуючий блог пов'язаний із конкретною темою, припустимо з професією, то знайти однодумців, друзів, колег буде значно легше. Блогери часто спілкуються між собою. Вони часто проводять різноманітні спільні конкурси та марафони, що приводить до взаємного піару та обмінами підписників. Так здійснюється обмін досвідом і підписниками між ними.
- Блог-портфоліо. Менеджери з різних компаній отримують кожного дня безліч нових резюме кожного дня. На мій погляд блог значно полегшить життя людям і допоможе якось виділитися із купи людей. Слід відмітити, що сам блог можна використовувати, як портфоліо, тобто завантажити туди свої проєкти, пропозиції, стартапи.
- Самопрезентація. Вона допоможе розвинутися у будь-які професії. Вдала самопрезентації може сформувати новий особистий бренд який буде вказувати на ваш професіоналізм.

- Нові навички. Розвиток блогу – це удосконалення своїх навичок. Саме очевидним буде те, що ви почнете краще та грамотніше писати. Слід відмітити, що блоги грають не аби-яку роль у навчанні студентів та що там навіть досвідчених спеціалістів. Коли є запитання чи деякі ідеї, ми всі починаємо шукати інформацію, людей з досвідом, які зможуть чітко пояснити певну тему. Мікроблоги на якусь конкретну тему зможуть допомогти, закріпити та доповнити знання, адже кращий спосіб щось усвідомити – це можливість пояснити деяку тему простою мовою, тобто так, як бачиш саме ти.

Слідуючи з розглянутої інформації було вирішено розробити універсальний веб-додаток, що являється блогом. Який в свою чергу має працювати набагато швидше за свої аналоги, цьому сприяє гнучкість інтерфейсу користувача, швидкість оброблення даних, конфіденційність даних а також можливість забезпечення повноцінної адміністрації та модерування даних.

РОЗДІЛ 1
АНАЛІЗІЗ ІСНУЮЧИХ ТЕХНОЛОІГЙ

1.1.Опис завдання

Програмна частина продуту була розділена на 2 частини: клієнтська та серверна. І це відразу дає зрозуміти свою перевагу, так як можлива динамічна зміна клієнтської частини не змінюючи серверну.

Веб додаток моделі загального публікаційного блогу було спроєктовано за для полегшення перегляду новин чи спостерігати за блогами своїх друзів, а також обговорювання питань які будуть темою даного блогу. Даний додаток надає можливість авторизованому користувачеві переглядати нові пости і якщо вони йому до вподоби він може поставити уподобання під ними, та створювати свої статті, а також є приємна функція оформити підписку на знайомих, а також людей, які зацікавили і переглядати їхні пости.

Для реєстрування не авторизований користувач моє надати свою електронну адресу на яку буде надіслано запрошення для підтвердження особи.

Авторизований користувач може змінити чи видалити написану ним статтю. Також у додатку присутня можливість зміни своїх особистих даних: пароль чи пошту. User може переглядати свої пости а також людей на яких він підписаний і які підписані на нього. Присутня дуже зручна функція пошуку блогу по тегу, яка допоможе вам найти цікаві статті в яких ви зацікавленні. Адміністратор має ті ж самі можливості, як і звичайний користувач але у нього є так деякі переваги і додаткові можливості. Отже однією з самих головних переваг є те, що він може редагувати і видаляти пости інших користувачів, а також у разі необхідності додавати модераторів, які будуть відслідковувати контент за для забезпечення всіх моральних та етичних норм.

1.2.Огляд існуючих програмних рішень

Було вирішено розглянути декілька найпопулярніших аналогів за для визначення конкретних цілий при розробці програмного продукту. Також ми

розглянемо усі переваги та недоліки, всі за і проти. Після аналізу буде вирішено, що буде використовуватися.

1.2.1. Facebook.com

На кінець 2019 року Facebook – є найпопулярнішим українським засобом для обміну інформації. Кожного дня на порталі facebook.com на вкладку «Новини» надходить більше півтори мільярда новин різного характеру із різних джерел світу. Серед них є безліч популярних сайтів, блогів як українців, так і міжнародних. Це можуть будь регіональні онлайн ресурси, різні інформаційні агентства, а також ЗМІ з усього світу. Нові новини оновлюються щосекунди і публікуються по заданій тематиці відносно уподобань користувача «Новини в Україні», «Спорт», «Політика», «Технології», «Друзі».

Модератори порталу facebook.com контролюють всі публікації, які надходять в стрічку новини і виключають матеріали які порушують морально-етичні норми, публікації для дорослих, публікації які містять погрози до життя користувачів, інформацію відносно приватного життя чи загрожують розвитку дітей.

Facebook це не тільки портал для перегляду новин. Це мережа в які користувачі можуть вільно і приватно обмінюватися інформацією, враженнями чи простими новинами між

Таким чином користувач, який використовує соціальну мережу facebook.com може вибирати, обговорювати і ділитися з друзями новинами, які йому до вподоби і видаються цікавими.

Інтерфейс Facebook.

Новому користувачеві першою відкривається головна сторінка[1]. У якій відображається функціональне меню, а також тут розташована стрічка новин, що дозволяє відстежувати події в групах і у друзів. Основні її елементи зображені на Рис. 1.1.

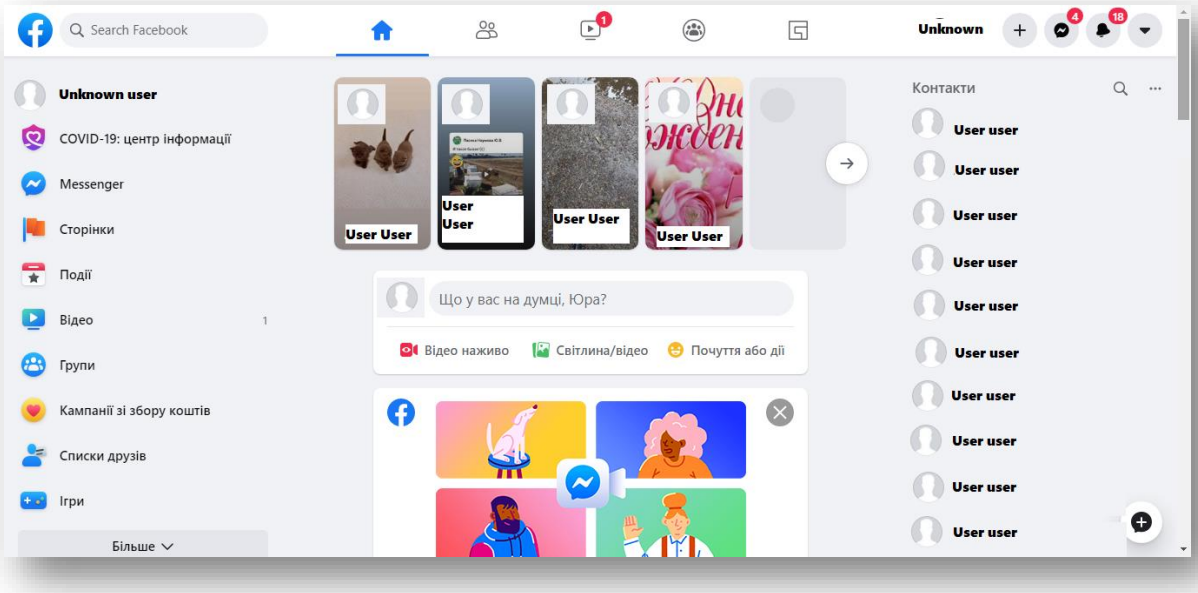


Рис. 1.1. Основний функціонал веб-сайту

Дослідивши Рис. 1.1. можна виділити основні переваги користувацького інтерфейсу:

1. Пошукова стрічка зображена на Рис. 1.2.- дозволяє швидко знайти публікацію, сторінку людини за ключовим словом.;

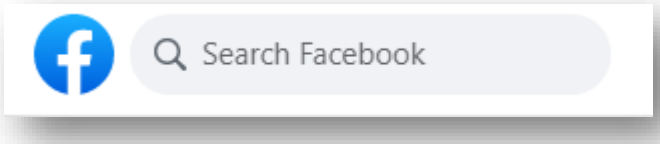


Рис. 1.2. Пошукова стрічка

2. останні події зображені на Рис. 1.3. - відображає усі отриманні повідомлення і події, котрі сталися, такі як: отримання особистого повідомлення, чи допису в якому ви відмічені, а також запит в друзі;

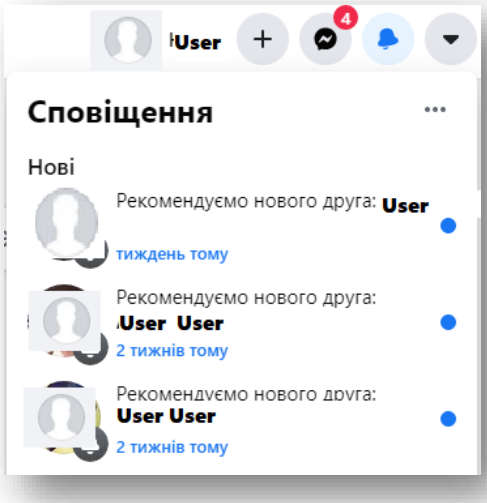


Рис. 1.3. Відображення останніх подій

3. додавання дописів «Що у вас на думці?» - дозволяє створювати власні дописи, ділитися з підписниками своїми думками, цікавими новинами, фотографіями;
4. стрічка новин – показує усі доступні оновлення груп, публікацій на сторінках друзів чи груп, на яких ви підписані;
5. чат – окрема програма, яка призначена для обміну особистими повідомленнями.

Серед переваг даного веб-додатку можна виділити наступні пункти[2]:

- Автоматичне оновлення та збирання новин від інших користувачів;
- Зручний обмін повідомленнями між користувачам.
- Можливість здійснювати відео дзвінків;
- Можливість використовувати рекламу для бізнесу;
- Додавання різноманітного контенту: фото, відео, чи звичайні статті.

Недоліками Facebook є:

- Невзаємність підписки. Тобто один користувач хоче відслідковувати новини іншого, але йому потрібно долучитися в друзі, а інший користувач його не знає і просто ігнорує запит в друзі;
- складний інтерфейс, для користувача який тільки перейшов на сайт;
- швидкість виконання запитів;
- адміністрація не встигає відслідковувати всі новини, тому дуже часто з’являється контент, який порушує права та конфіденційність даного веб-додатку.

1.2.2. YouTube

YouTube – це веб-додаток створений компанією Google і він інтегрований під всі пристрої і ОС, персональних комп'ютерів, телефон, планшет. На сьогоднішній день цей додаток займає друге місце серед різноманітних сервісів для перегляду контенту. Про це свідчить, що 79% користувачів інтернету зареєстровані на сайті. Він був створений для обміном самих різноманітних відео, це можуть бути «Новини», «Музика», «Спорт», «Політика» «Навчання», чи просто розважального контенту. Головна сторінка YouTube зображена на Рис. 1.4.

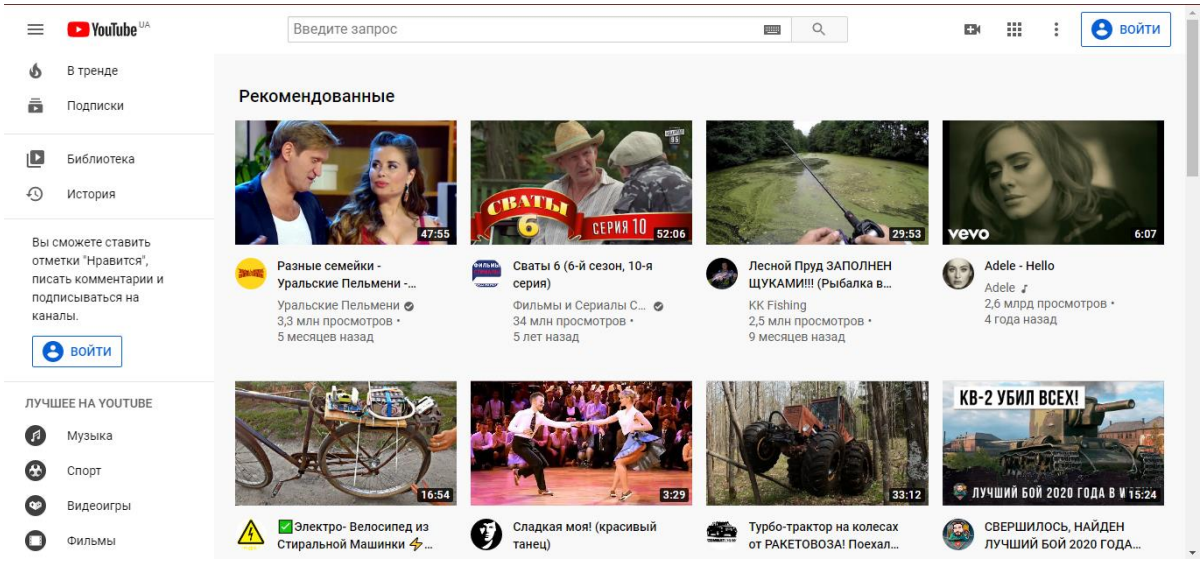


Рис. 1.4. Ілюстрація головної сторінки веб-додатку YouTube

Статистика показує, що щогодини завантаженні відеоролики переглядають мільйони користувачів[3]. У кожного користувача є можливість створити свій власний канал, який може розвивати публікуючи власні відео різного вмісту.

Непомітно для викладачів та батьків даний додаток відіграє неабияку роль в освіті учнів та студентів, та що там навіть у людей які працюють. Практично більша частина студентів використовують YouTube за для навчання, і це відіграє дуже важливу роль в освіті. Даний відео сервіс раніше вважався лише за для перегляду музичних кліпів і спостереженням за життям блогерів, та розважальних відео. Але на даний час він здійснює неабиякий

вплив і на освітню галузь науки. Використовуючи YouTube можуть підвищувати кваліфікацію і вчителі, розказуючи кожного разу щось нове, тим самим заохочуючи учнів новими темами для обговорення.

Коли користувач завантажує відео на YouTube і за замовчуванням відео встановлюється загальнодоступним, а це означає , що кожен може переглянути його. Кожне завантажене відео на порталі YouTube проходить перевірку на дотримання всіх норм, які затвердженні в правилах. І медіа будуть заблоковано якщо в них є порушення даних правил: зображення оголених людей або вміст сексуального характеру, шкідливий або небезпечний вміст, вміст, який пропагує ненависть, спам, оманливі метадані, шахрайство та інших.

Який основний функціонал YouTube?

- Пошук та перегляд відео;
- створювати персональний канал на YouTube;
- завантажувати медіа на свій канал;
- можливість залишити Уподобання/Коментар/Ділитися відео;
- можлива підписка/відписка;
- створення списків відтворення, щоб організовувати відео та групувати їх разом;

YouTube - це безкоштовний у користуванні сервіс, і він може стати прекрасним місцем для підлітків для виявлення речей, які їм подобаються. Для багатьох молодих людей YouTube використовується для перегляду музичних відео, комедійних шоу, інструкцій, рецептів, хакерів тощо. Також підлітки використовують послугу відеообміну, щоб слідкувати за своїми улюбленими влоггерами (відео-блогерами), підписатися на інші YouTube аккаунти та знаменитостей, які їх цікавлять.

Серед переваг YouTube слід виділити наступне[4]:

- Автоматичне оновлення головної стрічки відносно ваших уподобань;

- зручний і зрозумілий інтерфейс. Вигляд навігаційної панелі YouTube можна побачити на Рис. 1.5.

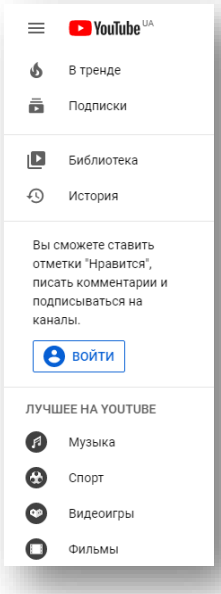


Рис. 1.5. Ілюстрація навігаційної панелі YouTube

- можливість використовувати рекламу за для розвитку вашого бізнесу;
- зручний плеєр для відтворення відео, можемо побачити його на Рис. 1.6.

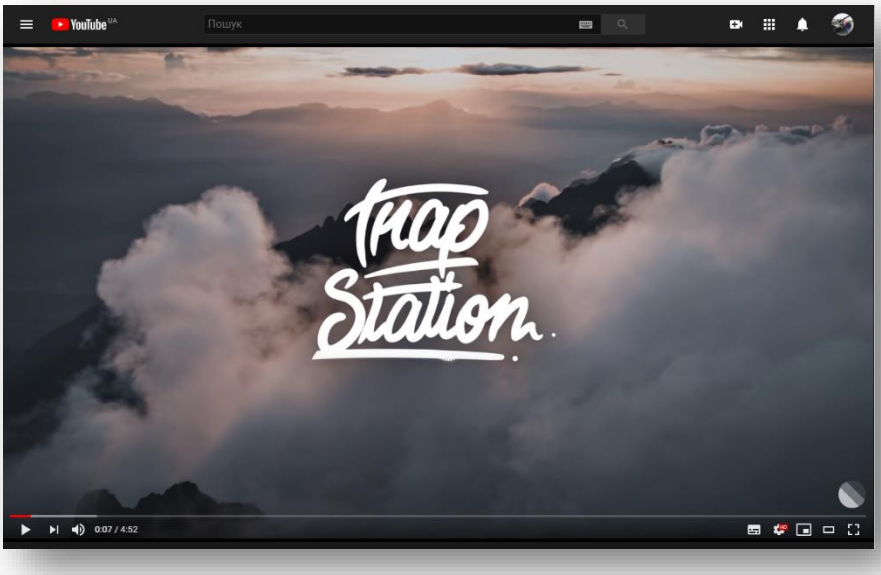


Рис. 1.6. Ілюстрація плеєра під час відтворення відео YouTube

- Різноманітність контенту;
- Можливість залишати коментарі.
- Створення і розвивання свого каналу.

Недоліками:

- Швидкість роботи;
- надання превілегій за реальні кошти;
- безліч реклами, яка заважає спокійно користуватися даним додатком;
- на мою думку найбільшою проблемою є відсутність чату;
- щоб поділитися відеом з друзями потрібно, тільки надати їм посилання;
- тяжко обмежувати контент для дітей.

1.2.3. Twitter

Twitter - це сервіс мікроблогів, який дозволяє відкрити свій власний мікро блог і писати туди короткі повідомлення. Кожне написане текстове повідомлення називається твіт. Розмір такого твіта обмежений і являє собою лише 140 символів, включаючи пробіли[5]. Twitter - це зростаюча мережа з відмінною перспективою на майбутнє, так, як зараз людям дуже подобається ділитися своїми враженнями і емоціями. Коли користувач робить твіт в свій мікроблог, він ділиться частинкою свого життям з усіма людьми, які на нього підписані.Ваші твіти будуть видним користувачам тільки, якщо вони підписані на ваш мікроблог. Підписників в Твіттері по іншому називають фоловерами. В свою чергу ви також можете підписувати на мікро блоги інших людей і бачити їх твіти. Також дуже зручна можливість писати твіти конкретній людині. Якщо задуватися, то Твіттер являє собою чат в якому ви бачити всі повідомлення людей, на яких ви підписані. Головну сторінку даного додатку ви можете бачити на Рис. 1.7.

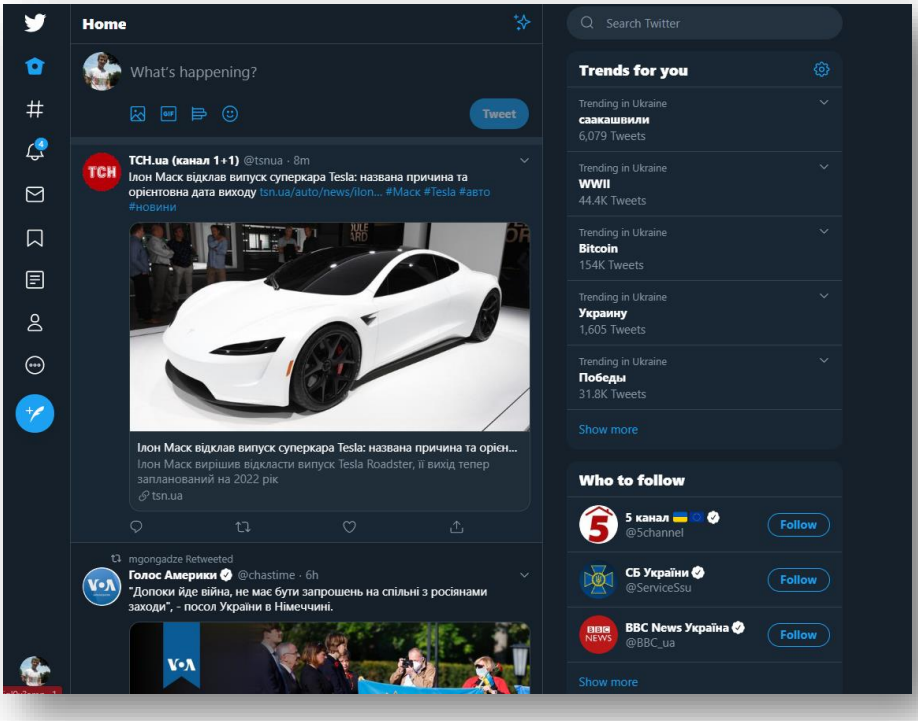


Рис. 1.7. Головна сторінка Твіттера.

Що до переваг Твіттера[6], то одна із них – це пошук за ключовими словами, що дозволяє побачити твіти, написані лише кілька секунд назад будь-яким із користувачів Твіттера, незалежно від того, підписані ви на нього чи ні. Жодна пошукова система у світі, будь то Гугл, Яндекс, чи будь-що інше, не забезпечить такої швидкості. Якщо вас цікавлять новини, пов'язані з Microsoft, ви просто набираєте в пошуку Твіттера "apple" і отримуєте сторінку з твітами про Apple, які оновлюється в режимі реального часу зі швидкістю десятки твітів в хвилину. Як тільки хто-небудь з десятків мільйонів користувачів Твіттера згадає в своєму твіті Apple, ви будете про це знати вже через лічені секунди.

Глобальність Твіттера обумовлена тим, що цей сервіс популярний у всьому світі. На відміну від "Однокласників" або "ВКонтакте", які популярні лише на території країн СНГ і в яких ви не знайдете ні одну зарубіжну знаменитість, крім фейків, в Твіттері ви можете бачити думки світових знаменитостей.

Одна з найсильніших сторін Twitter - функціонал, що дозволяє відстежувати тренди. Представленні актуальні теми для вас в вкладці Trents зображені на Рис. 1.8.

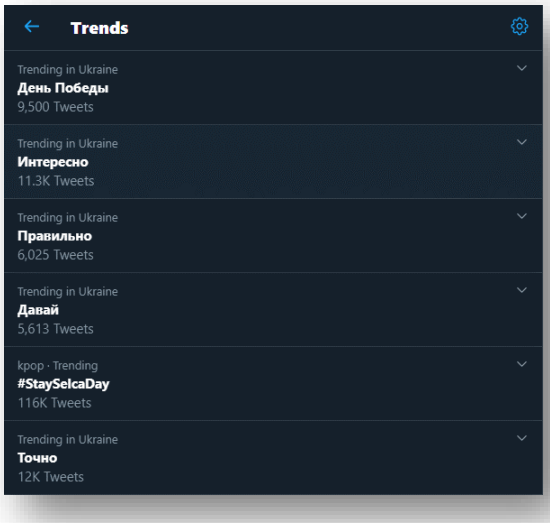


Рис. 1.8. Сторінка трендів в Твіттері.

Функціонал Twitter Trends є налаштовуючим – і його можна змінювати і налаштовувати на свій смак. Слід зауважити, що веб-додаток Twitter має дуже мало функціоналу. Наприклад, якщо ви маєте бізнес і плануєте його розповсюджувати на території України, то можна налаштувати тренди так, щоб ви дізнавалися інформацію про даний продукт один із перших.

ВИСНОВКИ ДО РОЗДІЛУ 1.

Протягом першого розділу було розглянуто аналогічні веб додатки такі, як Facebook, Youtube, Twitter. Ці додатки є по своєму особливі і креативні зі своїми плюсами та мінусами. Кожен з них це в певній мірі блог, у якому люди можуть вільно обмінюватися певною інформацією. Слід відміти, що було проаналізовано самі сильні та навпаки слабкі сторони представлених аналогів.

Проаналізувавши наявні популярні веб-аналоги з подібною функціональністю можна виділити набір певних вимог за для вдалого і ефективного створення свого програмного забезпечення:

- Наявність необхідного функціоналу, для швидкого обміну інформації;
- створення user-friendly інтерфейсу для зручного користування веб-додатком;
- дотримання архітектурних шаблонів для полегшення підтримки проєкту та швидкодії веб-сервісу;
- дотримання конфіденційних норм для збереження приватності та захисту персональних даних;
- підтримка мультимедійних файлів.

Виконавши запропоновані вимоги можна створити програмне забезпечення, який задовольняє примхи користувачів і буде зручним інструментом за для швидкого обмінну інформацією.

РОЗДІЛ 2
АНАЛІЗ ТЕХНОЛОГІЙ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ

2.1.Завдання та технології

Для даного програмного забезпечення, яке було розроблене в даному дипломному проєкті, потрібно обрати мову програмування для реалізації клієнт-серверної частини, також необхідно визначитися з фреймворками, які будуть покращувати і допомагати у розробленні даного продукту. Крім мови програмування та фреймворків необхідно обрати СКБД для збереження даних.

На даний момент існує різноманітність і варіативність вибору мов програмування та фреймворків для розроблення свого власного програмного продукту. На разі в ІТ індустрії переважно використовується клієнт-серверна архітектура.

Дана архітектура є одною із архітектурних шаблонів програмних продуктів та на даний час саме вона переважно використовується у розробленні застосунків. Клієнт-серверна архітектура передбачає собою взаємодію та обмін даними між ними, а саме:

- Ряд серверів, які надають певну інформацію та сервіси програмному забезпеченню, яке надсилає запит до них;
- Певний набір клієнтів використовують наді їм сервіси, які в свою чергу надані їм серверами.
- І мережа без якою не можлива взаємодія між клієнтом та сервером, переважно це Інтернет.

Чому дана архітектура користується популярністю? А ось чому. Сервери, як і клієнти не залежать один від одного і працюють паралельно. На Рис. 2.1 ми бачимо вигляд даної архітектури.

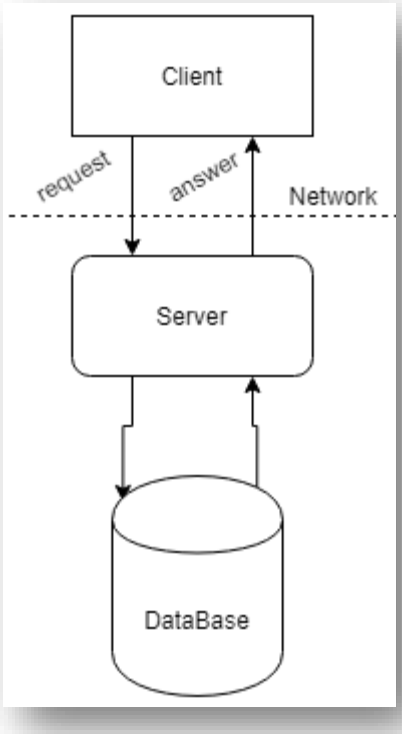


Рис. 2.1. Клієнт-серверна архітектура

Якщо детальніше розібратися з зображенням рис 2.1, то можна розділити веб-додаток на 3 частини:

1) Клієнт – програмна частина даної архітектури, що відповідає за безпосередню взаємодію з серверною частиною і дозволяє отримувати певну інформацію від неї.
2) Сервер – програмна частина, що дозволяє обробляти певну інформацію і надсилати усім клієнтським застосункам, що підписались на її оновлення.
3) База даних – програмна реалізація технічного рішення для зручної структуризації необхідних даних.

В наступних розділах буде досліджено актуальні технології, за для якісного забезпечення клієнт-серверної архітектури.

- Клієнтська частина: Thymeleaf, Node.js, Bootstrap
- Серверна частина : C#, Python, Java

- СКБД: PostgreSQL, MySQL, MongoDB

Після розгляду і аналізу досліджень буде обраний найкращий набір технологій для подальшої реалізації розроблюючого продуту.

2.2. Аналіз та вибір мови програмування

Пропоную розглянути найпопулярніші мови програмування та їх фреймворки для розробки програмного продукту.

2.2.1. Python

Python – високорівнева мова програмування якій притаманні такі властивості, як об'єктно-орієнтованість, інтерпритованість, яка містить динамічну семантику[7]. Високо рівневі вбудовані структури даних у поєднанні з динамічним набором тексту та динамічним зв'язуванням роблять її дуже зручною для швидкого розвитку додатків, а також для з'єднання існуючих компонентів разом. Окрім всього переліченого важливим є те, що в даній мові програмування притаманна підтримка декількох парадигм програмування, таких, як функціональність, об’єктно-орієнтованість чи процедурність.

Часто програмісти закохуються в Python через підвищену продуктивність, яку він забезпечує. Оскільки кроку компіляції немає, а цикл редагування-тестування налагодження надзвичайно швидкий. Також він доволі активно і швидко розвивається, що зумовлюється рентабельними та актуальними оновленнями з боку розробників. Додаються та покращуються вбудовані в дану мову програмування функції, але смисл мови не змінюється. Для розробників одна із чудових можливостей є те, що програмний код написаний на Pyton є легкий та зрозумілий новачкам на проєкті.

Самими яскравими особливостями, які дуже часто використовуються є:

- Динамічна перевірка типів та типізація;
- автоматичне управління пам’яттю;
- динамічна перевірка типів;

- паралельність виконання;
- оброблення виключень.

По замовчанням в дану мову програмування вбудована стандартна бібліотека, яка величезний функціонал та набір корисних функцій, які були написані раніше, окрім цього всього також вбудованні і колекції такі, які множини, списки, та інші.

Інтерпретатори мови Python доступні на багатьох операційних систем таких як: Windows, MacOS, Linux. На даний момент актуальна версія даної мови програмування – Python 3.x. Не беручи до уваги динамічну типізацію, то Python є сильно типізованою мовою, тобто компілятор забороняє операції, не чітко визначенні, наприклад додаючи рядок до дробового числа, на що видає спеціальну помилку.

А що до наслідування, то в Phyton притаманне як одиничне, так і множинне наслідування. Також є можливість наслідування від вбудованих типів і їх розширення, щоб досягти чистої реалізації складних шаблонів проєктування використовують мета програмування. Зокрема слів відмітити, що мова Pyhon є однією із найпопулярніших у світі на сьогоднішній день, що в свою чергу відслідковується за кількістю існуючих фреймворків, які щоденно спрощують наше життя за рахунок використання уже готових класів та функцій, котрі сприяють розширенню програмного коду.

Найпопулярніші фреймворки в даній мові програмування зображені на Рис. 2.2.

Рис. 2.2. Найпопулярніші фреймворки Python

Ось декілька фреймворків:

1) Django. Даний фреймворк використовує застосунок «shared-nothing» архітектуру, яка представляє з себе, що кожна з частин архітектури не буде залежати від інших, тож може бути змінюватися або замітися при виникненні необхідності.
2) Flask. Це такий мікро фреймворк, як Django і переважно використовується при розробленні веб-додатків.

Представленні фреймворки є найпопулярніші серед веб-додатків на мові програмування Python, і не задарма, але вони вагомо відрізняються друг від друга по своїй логіці.

Ось наприклад Django буде поставлятися з ORM за для SQL баз даних, що надає вам безпосередній функціонал для запуску екземпляра бази даних. В той час як Flask для запуску бази даних вимагатиме додаткових кроків.

Python ряд переваг та недоліків

До переваг слід віднести:

- Швидкість;
- повна комплектація;

- безпека;
- масштабованість;
- різнобічність.

А що до недоліків то:

- із статичним URL, який був зазначений проходить шаблон маршрутизації;
- фреймворк Django не розбитий по архітектурі;
- все основана на orm django;
- необхідно вміння володіти всією системою для роботи;

2.2.2. Java

Java – це об’єктно-орієнтована програмування мова яка призначена для розробки програмного забезпечення загального призначення. Вона реалізована за допомогою класів та об’єктів. Передусім вона розроблялась як крос-платформна мова програмування, тому вона містить значно менше низькорівневих можливостей для роботи з системним обладнанням, в порівнянні наприклад C++. При необхідності таких дій в Java є можливість виклику підпрограм, які були написані сторонніми мовами програмування. Також слід відмітити, що створення даної мови програмування зумовлено тим, щоб позбутися глибокої залежності коду. Крім цього слід зауважити, що Java - повністю об'єктно-орієнтована мова програмування, навіть C++ стоїть на ряд нижче. Переконуючим фактом є те, що всі сутності в даній мові є об’єктами окрім небагатьох типів.

Програмне продукт, який написаний на мові Java не залежить від архітектури програмного забезпечення на якому буде запускатися даний програмний код, так як розроблююче програмне забезпечення компілюються до байт коду[8]. При виконанні програмного коду на мові програмування Java, завдяки віртуальній машині, він інтерпретується для конкретної платформи.

При розробленні даної мови програмування було взято за основу синтаксис із мов програмування C та C++. Слід відмітити, що також було запозичено об’єктну модель із мови C++, проте з плином часу вона піддалася глобальній модифікації. Враховуючи конфліктні ситуації, які можуть часто виникнути через помилки програмістів, їх було усунуто. А сам процес розробки програмного забезпечення на даній мові був значно полегшений. Ряд необхідних дій, які були важливі в C++ в Java виконуються за допомогою віртуальної машини.

У мові програмування Java присутній різноманітний вибір додаткових фреймворків, які в свою чергу значно полегшують розроблення програмного продукту. Статистика фреймворків, які найчастіше використовуються програмістами наведена на Рис. 2.3.

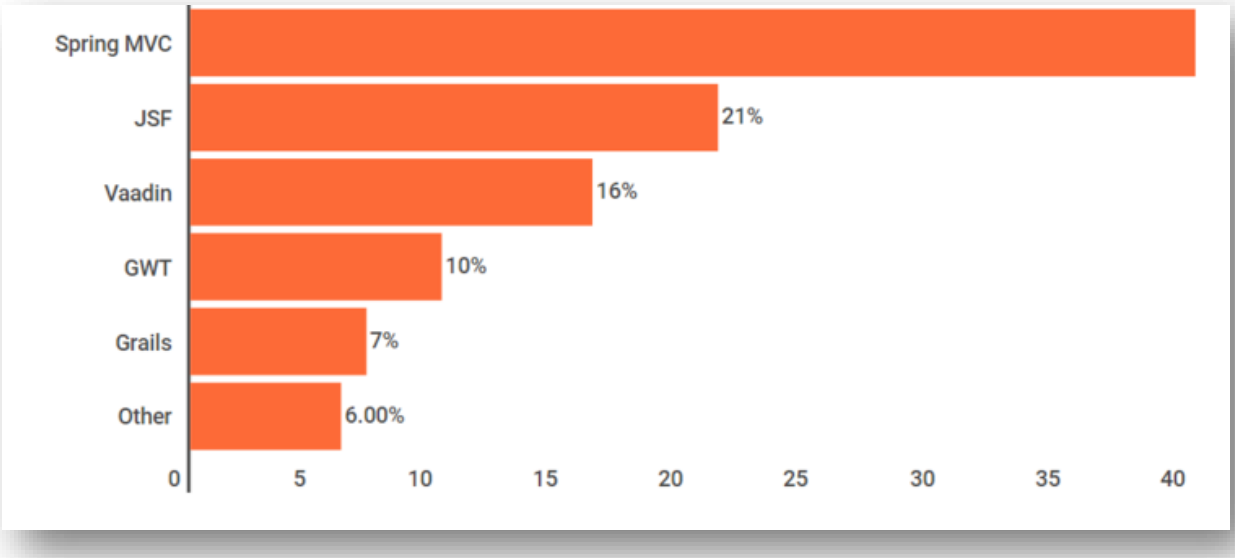


Рис. 2.3. Статистика фреймворків

Пропоную розглянути і дослідити Spring MVC. Фреймворк Spring – це один з самих популярних фреймворків для створення веб-додатків на Java, але слід відзначити, що використовуючи даний фреймворк є можливість не тільки створювати веб-додатки, але і для самих звичайних консольних програм.

В кожній новій версії Spring намагаються відійти від жорсткої зв'язності, це коли написані вами класи безпосередньо мають залежність від інших класів чи інтерфейсів з даного фреймворка. І щоб не виникла дана залежність використовують для цього анотації. Слід зауважити, що Spring це лише набір написаних для вас класів і інтерфейсів, які будуть полегшувати вам життя.

А що до структури Spring MVC слід відмітити, що це не лише один якийсь конкретний фреймворк. Краще сказати що це загальна назва для ряду фреймворків, кожен з яких виконує свою роботу[9]. Структуру даного фреймворку ми можемо бачити на Рис. 2.4

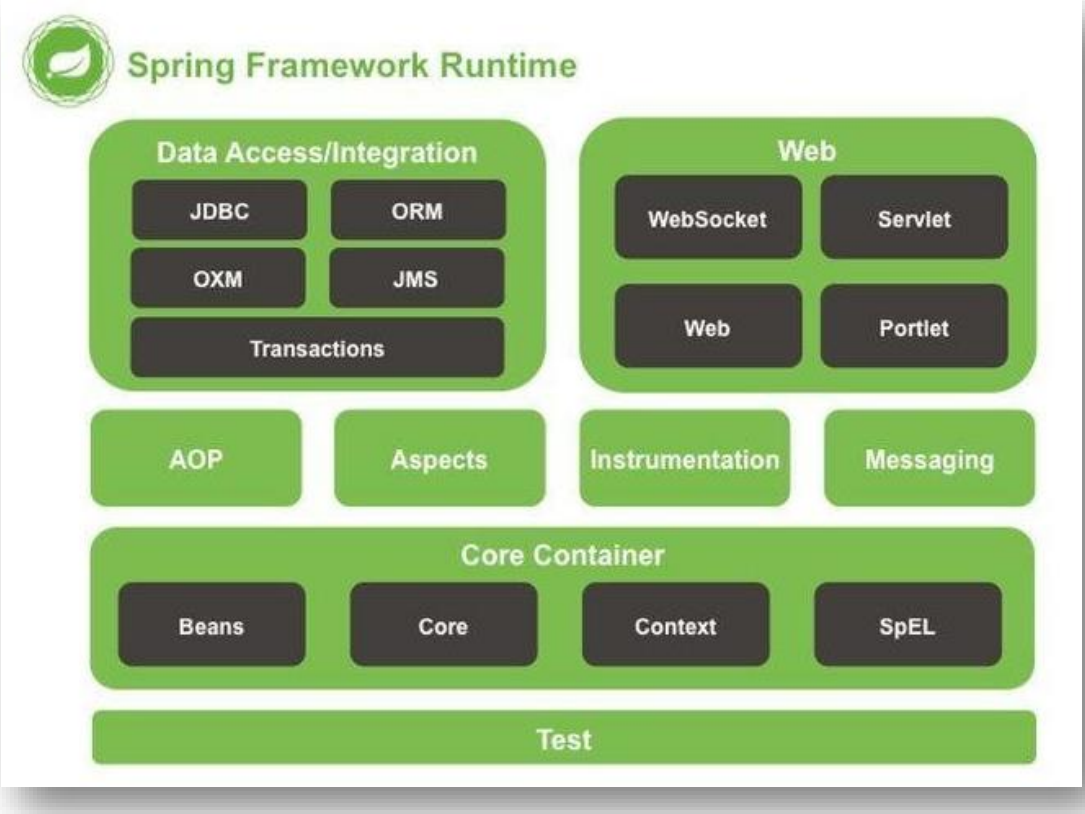


Рис. 2.4. Структура Spring Framework

Дослідивши дану мову програмування ми можемо визначити наступні переваги:

- Об’єктно-орієнтована мова;
- строга типізація мови;
- синтаксис мови. він дуже схожий на синтаксис мови програмування с та с++;
- безпека;
- різноманітний вибір фреймворків;
- крос-платформеність;
- динамічність;
- перевірка допущених помилок на етапі компіляції, що дозволяє заощадити час розробки програмного забезпечення.

Серед недоліків слід виділити наступне:

- Продукт на Java потребує багато пам’яті;
- потребує написання великої кількості коду, що в свою чергу сприяє виникненню помилок;
- швидкість. слід відмітити, що дана мова програмування повільніша в порівнянні з іншими мовами.

2.2.3. C# і Microsoft.NET Core

C# – об’єктно-орієнтована мова програмування, яка має статичну типізацію. В той час, коли дана мова програмування тільки розроблялася команда розробників опиралися на існуючі рішення такі як Java, C++, Modula. І за допомогою саме тих рішень удалось уникнути ряду проблем, ось яскравий приклад, множинне успадкування у C++.

C# базується на технології Common Language Runtime(CLT) – загальномовне виконуюче середовище. Тобто дана мова програмування має технологію CLT так як Java - JVM. C# також крос-платформна мова програмування завдяки того, що код програми транслюється у проміжний байт-код, що в свою чергу дозволяє писати додатки для різних платформ, не міняючи

код самої програми. Для створення додатків клієнт-сервер, додатків баз даних дуже зручно використовувати дану мову програмування. Середовище для розробки Visual C# надає розширений функціонал редагування для редагування програмного коду та багато інших засобів, які покращують розробку веб-додатків на мові C# для платформи .NET Core.

Синтаксис C# є дуже подібним до мов C, C++ та Java[10]. Також дана мова програмування реалізовує статичну типізацію. Слід відмітити, що керування пам’яттю контролюється автоматичним збирачем сміття, який був реалізованим в CLR.

Слід зауважити, що C# дає можливість використовувати ряд мовних конструкцій, які будуть спрощувати розроблення нового програмного забезпечення та їх компонентів:

- інкапсульовані сигнатури методів, іменовані делегатами, які дозволяють реалізувати безпечне для типів повідомлення про події;
- атрибути, що надають декларативні метадані про типи під час виконання;
- в XML-документації дозволенні внутрішні кометарі;

Одними з основних переваг C# :

- Використання ряду бібліотек і шаблонів, які допоможуть вам у розробці;
- дана мова програмування використовує об’єктно орієнтований підхід;
- об'єднання кращих сучасних мов програмування: Java, C ++, Visual Basic та ін.;
- Різноманіття існуючих фреймворків. На Рис. 2.5. можемо бачити основні фреймворки в C#.

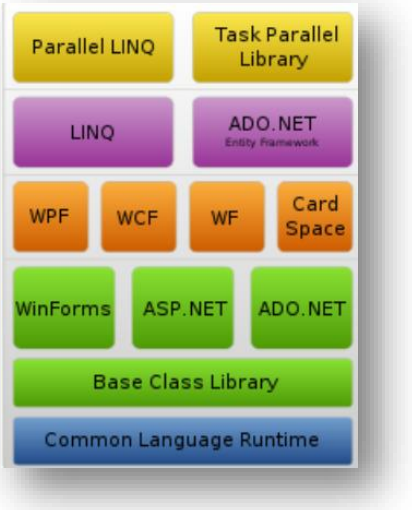


Рис. 2.5. Основні фреймворки C#

Пропоную розглянути фреймворк ASP.NET так, як він самий ефективної для веб-розробки. Даний фреймворк розроблявся для створення веб-сервісів та веб-застосунків. Слід відмітити, що ASP.NET є найбільшим для створення веб-сервісів та веб-додатків. Веб-сторінки написані за допомогою даної технології, відомі як веб-форми, є головними елементами для розроблення додатків. Існують дві основні методології веб-форм – це формати веб-додатків та веб-сайтів. В даному фреймворку присутня підтримка методології MVC, що в свою чергу приносить ASP.NET ще більшої популярності серед розробників. Можемо бачити архітектуру даного фреймворку на Рис. 2.6.

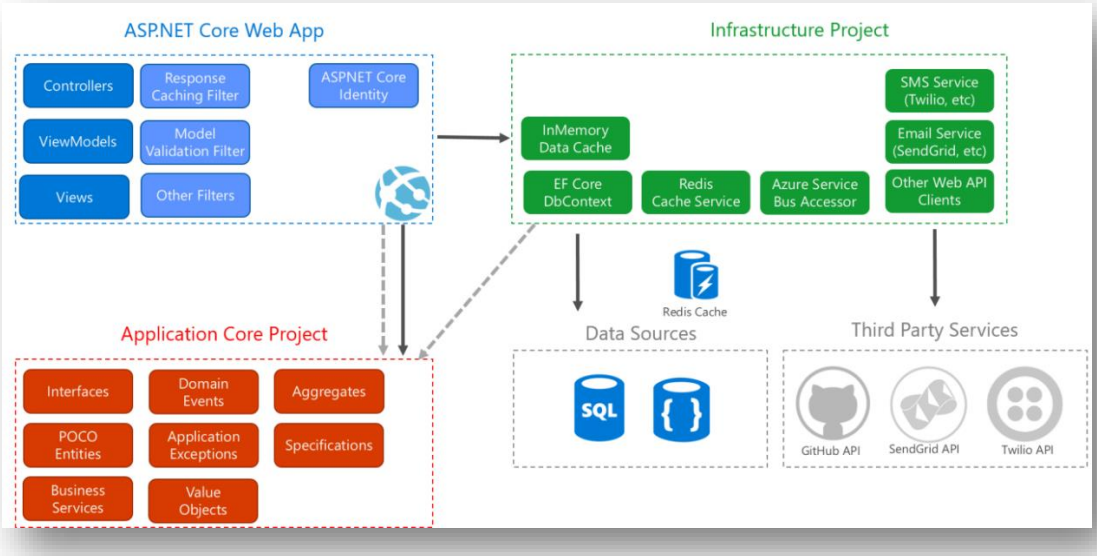


Рис. 2.6. Структура ASP.NET

Слід зауважити про недоліки C#:

- C # дуже легко дизасемблюється. Це означає, що з твій код буде отримано і вивчений конкурентами. Є спеціальні інструменти, які допоможуть ускладнити цей процес, але від цього захиститися не можливо;
- Пріоритетна орієнтованість на Windows платформу;
- В мові залишилася можливість використання оператора безумовного переходу.

2.2.4. Bootstrap

Bootstrap - це фреймворк, який містить об'ємний набір інструментів для швидкого створення зручних сайтів і веб-додатків. Даний фреймворк містить в собі HTML - і CSS - шаблони оформлення для веб-форм, міток, кнопок, блоків навігації та інших основних складових веб-інтерфейсу, включаючи розширення JavaScript. В Bootstrap використовуються сучасні технічні засоби в області CSS і HTML[15]. Використання даного фреймворку набуло своєї популярності не тільки серед незалежних розробників, але часто й цілими компаніями.

Основною областю його застосування – є розробка клієнтської частини сайтів. Серед аналогічних фреймворків Foundation, UIkit запропонований фреймворк є найпопулярнішим .

А популярність Bootstrapa пов'язана з тим, що він пришвидшує верстку сайтів, ніж це можна виконати на «чистому» CSS і JavaScript. А в даний час - це найдорожчий ресурс. Також його популярність пов'язана з доступністю . Вона полягає в тому, що на ньому навіть новачок може верстати досить якісні макети , які важко було б виконати без глибоких знань веб-технологій і достатньої практики.

Фреймворк Bootstrap є набір CSS і JavaScript файлів . Щоб його використовувати ці файли необхідно просто підключити до сторінці . Після підключення вам стануть доступні такі інструменти даного фреймворка як: колоночная система, класи і компоненти .Структуру даного фреймворку можемо бачити на Рис. 2.7.

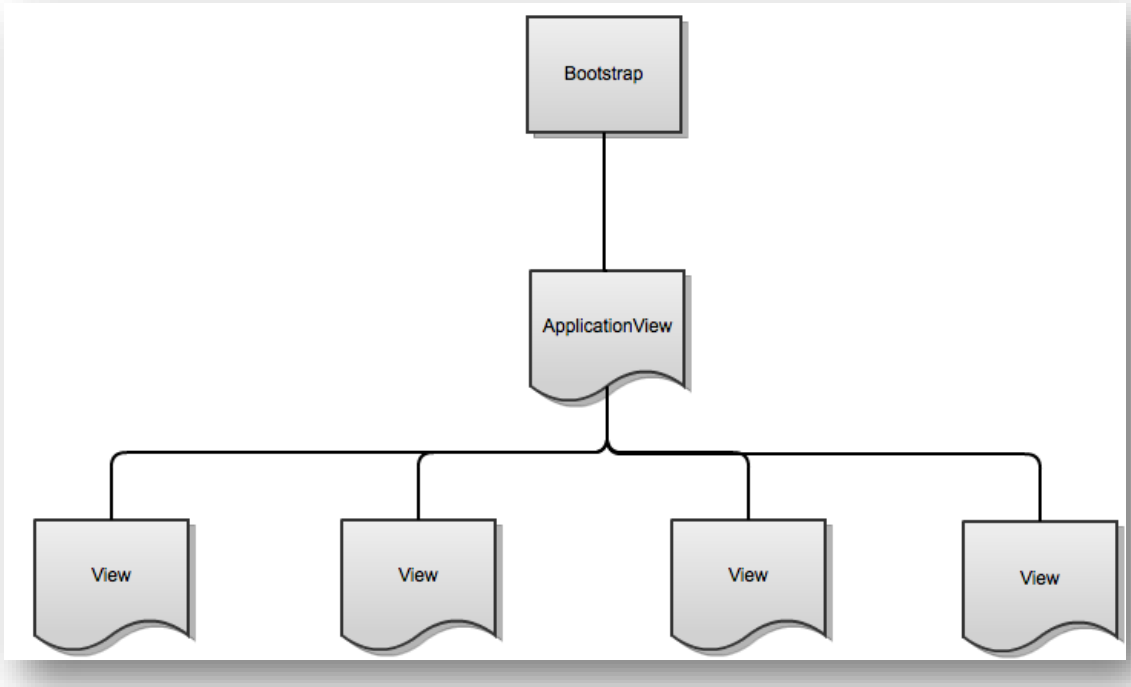


Рис. 2.7. Структура фреймворку Bootstrap

Перелік основних інструментів, яка необхідна для сучасної розробки сайтів:

- Таблиці. Створення і оформлення таблиць з елементами сортування;
- Типографіка. Опис і робота з будь-якими видами шрифтів;
- Навігація. Класи, що застосовуються для оформлення будь-яких навігаційних елементів сайту (наприклад, меню або вкладки);
- Алерт. Застосовується для створення спливаючих діалогових вікон або вікон з підказками;
- Медіа. Застосовується для управління відеозаписами і картинками;
- Шаблони. Установка по роботі шаблонів документів;
- Форми. Дані класи застосовуються для взаємодії з формами і їх подіями;
- Сітки. Установка і маніпулювання колонками;

Явне використання фреймворку Bootstrap зображено на Рис. 2.8.

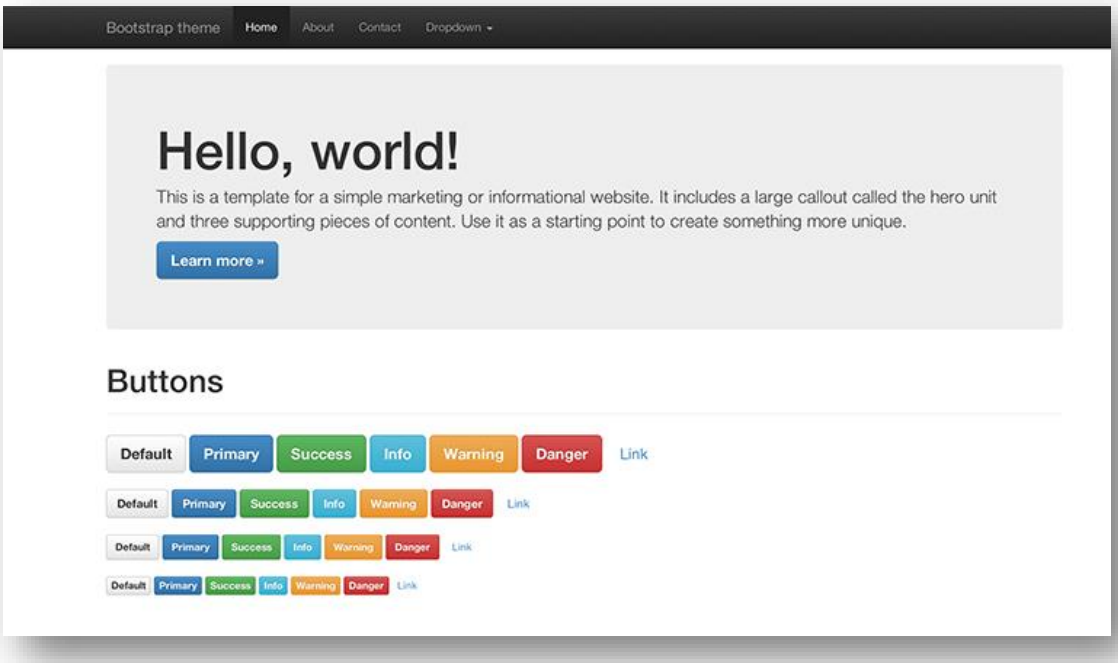


Рис. 2.8. Вигляд веб-сторінки з використанням Bootstrap

До позитивних сторін слід віднести наступні переваги даного фреймворку:

1. Наявність локалізації стилів;
2. Підтримка на будь-яких браузерах;
3. Великий набір шаблонів;
4. Висока швидкість роботи;
5. Простота у використанні.

Негативні сторони:

1. При роботі з JavaScript проявляються незначні баги;
2. Рідке виникнення конфліктів між інструментами;
3. Довготривале усунення багів даного фреймворку.

2.2.5. JavaScript (JS)

JavaScript – це об'єктно-орієнтована скриптова мова програмування, яка є діалектом мови ECMAScript. Дана мова програмування не призначена для створення автономних додатків. Програмне забезпечення, що було написане за допомогу мови програмування JavaScript вставляється в вихідний HTML код. І буде інтерпретоване браузером по мірі завантаження цього документа. Слід відмітити, що за допомогою даної мови програмування є можливість динамічної змінювати текст HTML-документа, а також реагувати на події, які можуть виникнути під час дій користувача, або зміна стану документа.

Важливою особливістю JavaScript є об'єктна орієнтованість[14]. Розробнику стає можлива можливість вільно працювати з об’єктами, наприклад: гіперпосилання, документи, форми та інші. Об'єкти характеризуються своїми властивостями і можливими методами, які можуть бути реалізовані. Слід відмітити, що він не надає низькорівневого доступу до процесора чи пам’яті так, які він був розроблений для браузерів, які не мають потреби у використання даного доступу. Можливості JavaScript серйозно

залежать від середовища, в якому він використовується. Пропоную розглянути Node.js.

Даний фреймворк підтримує функції, які дозволяють JavaScript виконувати запити мережі або читати чи записувати файли.

Використання мови програмування JavaScript надає можливості при яких користувач має необмежені можливості, що надає маніпуляцію між веб-сторінками, а також взаємодія з користувачем та веб-сервером.

Наприклад, JavaScript у браузері:

- Додати новий HTML документ на сторінку або змінити існуючий вміст веб-сторінки;
- Реакція на всі дії, які виконує користувач.
- Отримати та встановити файли cookie, задавати питання відвідувачеві, показувати повідомлення.
- Запам’ятайте дані на стороні клієнта («локальне зберігання»).

Зазвичай дану мову програмування використовують, як вбудовану мову програмування для програмного доступу до об’єктів заданих додатків. Частіше за все JavaSript використовується у браузерах, як мова, яка призначена для написання певних сценаріїв, що в свою чергу надає інтерактивності веб-сторінкам.

JavaScript основана на базі клієнта і складається з деяких загальних функцій програмування, які дозволяють виконувати такі дії, як:

- Зберігати корисні значення всередині змінних.
- Операції над фрагментами тексту
- Запуск коду у відповідь на певні події, що відбуваються на веб-сторінці.

Використовуючи дану мову програмування стають доступними такі функції:

- Зміна сторінок браузерів;

- додавання, зміна або видалення тегів веб-сторінки;
- зміна стилів веб-сторінки;
- запит до будь-якої частини вихідного коду програми;
- виконання різних дій з cookie-файлами.

Переваги JavaScript:

- Просте використання написаних;
- корисні функціональні налаштування;
- взаємодія з додатками може здійснюватися навіть у текстових редакторах;

Недоліки:

- Понижений рівень безпеки через вільний доступ до вихідного коду популярних скриптів.
- Безліч дрібних дратівливих помилок на кожному етапі роботи.
- Частина активно використовуваних програм (особливо додатків) перестануть існувати при відсутності мови, оскільки цілком базуються на ньому.
- Дана мова програмування містить ряд фреймворків, які вам допоможуть при розробленні програмного продукту. На Рис. 2.9. зображенні самі популярні фреймворки для JavaScript.

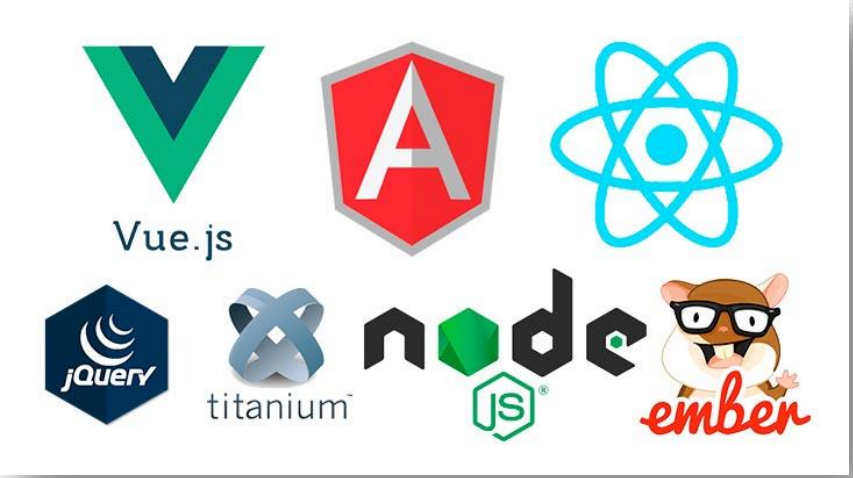


Рис. 2.9. Фреймворки JavaScript

Пропоную розглянути VueJS. Це фреймворк JavaScript, який має відкритий вихідний код. Він розроблявся за для розробки і проєктування користувацького інтерфейсу. Слід відмітити, даний фреймворк можна з легкістю інтегрувати у масштабні проєкти, тому що він практично завжди працює з рівнем візуалізації. VueJS добре розширюваний фреймворк і він немає залежностей чи великої кількості бібліотек. Основним завданням давного фреймворку є вирішення проблеми на рівні представлення, що спрощує інтеграцію з іншими бібліотеками в існуючих проєктах.

До переваг слід віднести:

- Невеликий розмір. За розміром він 18–21 Кб і користувач не витрачає багато часу на завантаження чи використання. Слід відмітити, що через малий розмір він не має низьку швидкість роботи;
- розробник має можливість легко додати даний фреймворк до свого веб-додатку, а це все тому, що він має просту архітектуру. Використовуючи дану схему є можливість розробляти проєкти різної складності.
- проста інтеграція;
- присутність реактивних даних;
- гнучкість при роботі з шаблонами;

- декларативний рендеринг;
- директиви
- Використання DevTools за для розширеного функціоналу браузера.

Недоліки:

- необхідність транслювання коду в ES5;
- відносно малий час на ринку, не надто велике товариство.

2.2.6. Вибір СУБД.

СКБД (система керування базами даних) – це додатки, які засновані на базах даних і варіантах застосування цих баз даних, це можуть бути: вибірки, редагування, видалення, вставки. Дана система розроблена за для безпеки і цілісності даних.

До основних функцій СКБД відноситься:

- Керування даними в області пам’яті;
- операції буфера;
- глобальна підтримка мови програмування SQL;
- керувати транзакціями;

Існують кілька варіантів баз даних і їх класифікують на

- реляційні;
- об’єктно-реляційні;
- об’єктно-орієнтовані.
- ієрархічні;
- мережеві;

Було обрано дві СКБД для порівняння – MySQL та PostgreSQL. Дані системи підтримують основні функції для керуванням базую даних.

MySQL і PostgreSQL - найпопулярніші і перевірені часом реляційні системи управління даними з відкритим кодом. На сам перед потрібно відзначити, що MySQL – реляційна, а PostgreSQL – об’єктно-реляційна. Обидві

технології використовують мову програмування SQL, щоб створити запити до баз даних чи таблиць. Що дозволяє безкоштовно їх використовувати.

Слід відмітити що, MySQL є швидшим ніж PostgeSQL, але менш функціональними так, як він забезпечений для дрібних та середніх проєктів. PostgreSQL - це повільніша, але більш функціональна СУБД і часто використовується на більш маштабних проєктах[13]. MySQL користується неабиякою популярністю в різному програмному забезпеченню завдяки своїй простоті і швидкості. PostgreSQL - це більш об'єднана база даних з одним механізмом зберігання[11].

Під час порівняння цих двох систем баз даних, необхідно вказати, який саме двигун буде використовувати MySQL, так як це значно впливає на функціональність, продуктивність та доступність. У MySQL досить часто використовується двигун InnoDB так, як завдяки ньому у даній СУБД є висока продуктивність з великою кількістю паралельних запитів але при незначних навантаженнях. У представлених СУБД є можливість бути оптимізованими відповідно до середовища розробки.

Дуже складно надати порівняння продуктивності даних СУБД, при умові що ми не зважаємо на конфігурації. У PostgreSQL і MySQL є можливість використання ряду технологій за для підняття продуктивності. Розробка MySQL відразу була спрямована до продуктивності, в той час, як стандарти PostgreSQL були спрямовані на акцент функціональності. Тож можемо вважати, що вибір MySQL вважається швидшим на дрібних і середніх додатках, в порівнянні з нею PostgreSQL більш надійним і швидшим на складних масштабних проєктах. На Рис. 2.10. зображено робочий інтерфейс бази даних PostgreSQL.

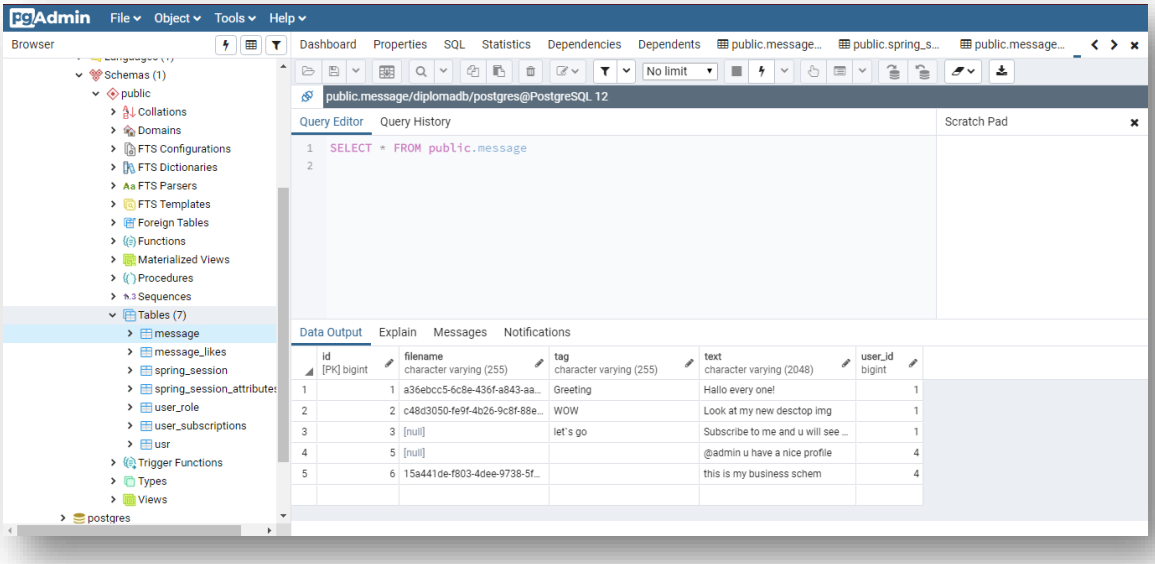


Рис. 2.10. Інтерфейс СУБД PostgreSQL.

Переваги PostgreSQL:

- Ефективне виконання статичних або параметризованих запитів SQL;
- розширений оптимізатор запитів;
- різні типи індексації: функціональна, часткова, сукупність декількох індексів;
- стиснення даних.;
- з версії PostgreSQL 8.2 реалізована масштабованість з записами навантаження;
- асинхронне виправлення.

Найчастіше за все СУБД PostgreSQL налаштовується тільки для роботи на веб-сервері і тому має слабку продуктивність. Є можливість підвищити продуктивність, але на сервері потрібно змінити декілька ключових параметрів конфігурації. У MySQL є можливість підтримка протоколу стиснення мережевих даних, який регулюється на стороні користувача. Дана функція передбачає стискання усіх даних, які переходять на сервер від користувача. Веб-розробники часто використовують MySQL через простоту

використання. Наразі MySQL став суттєво складним, в той час в PostgreSQL відбулися важливі зміни в структурі, то ж деякі розробники рекомендують використовувати його, так як він став простішим, ніж MySQL. На Рис. 2.11. можемо бачити архітектуру СУБД MySQL[12].

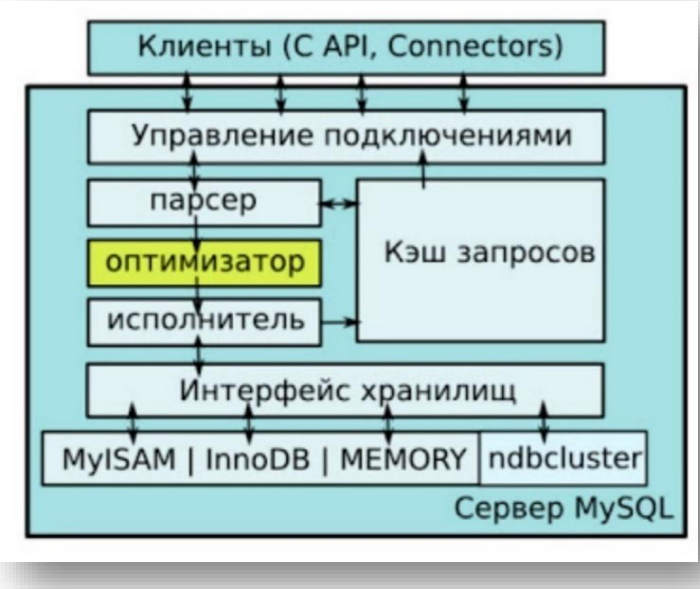


Рис. 2.11. Архітектура СУБД MySQL

Технологія InnoDB – це одна з ряду підсистем низького рівня в СУБД MySQL і наявна у всіх операційних системах. Слід зауважити, що основною відмінністю від інших низькорівневих підсистем - є наявність механізму транзакцій і сторонніх ключів для конфігурації. Формат даних в даній технології забезпечує блокування даних при трансакційних зв’язках, що забезпечує надійне зберігання даних. Дана технологія зберігає свої дані у великих спільних файлах в порівнянні з MyISAM, де файл з даними буде створюватися для кожної таблиці. На Рис. 2.12. зображено робочий функціонал MySQL WorkBench.

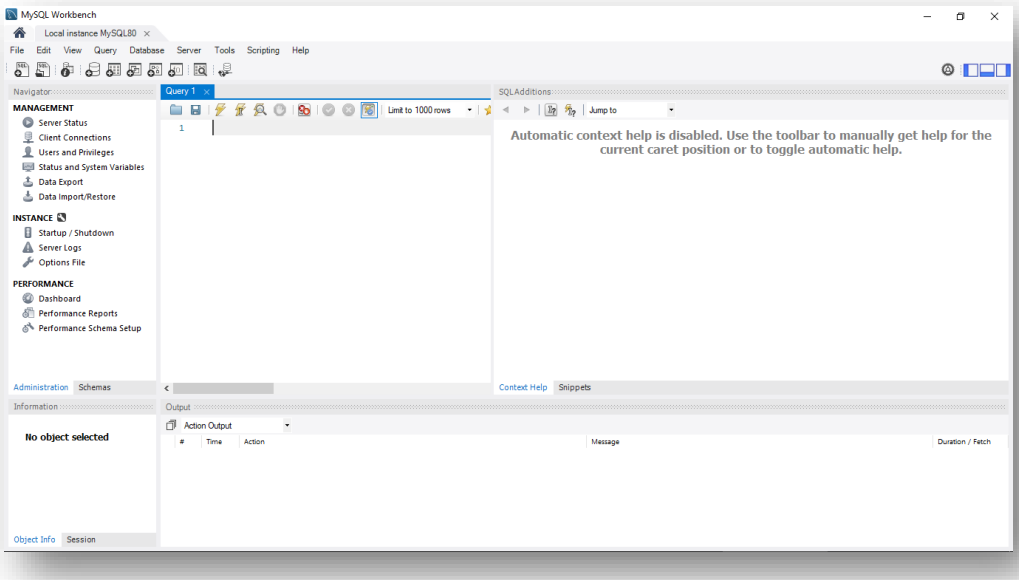


Рис. 2.12. WorkBench середовище для MySQL

Слід відмітити, що у даній технології база даних використовується тільки для збереження простих даних, тому немає потреби розробляти нову, більше складнішу базу даних з використанням рекурсивних запитів або реалізації розподілу. Дуже важливим аспектом для бази даних є швидкість прийому і запису інформації. Дивлячись на технічні завдання розроблюючого продукту, було прийнято рішення обрати базу даних PostgreSQL за для подальшого розвитку проєкту.

ВИСНОВКИ ДО РОЗДІЛУ 2

В даному розділі було розглянуто та проаналізовано набір сучасних технологій для написання програмного забезпечення. Дані технології мають не аби яку популярність і кожна з них рекомендує себе по особливому. Однак серед великого різноманіття сучасних технологій необхідно віддавати перевагу лиш тим, які найкраще підходять для вирішення конкретно поставлених завдань.

Одним із основних компонентів для структуризації даних було обрано базу даних PostgreSQL. Дана база даних дає усі необхідні можливості для структуризованого зберігання даних. Одним із основних переваг представленої бази даних являється можливість обробки великої кількості даних, що в свою чергу дає можливість розширювати проєкт, ігноруючи необхідність змінювати базу даних.

Для серверної частини системи, було прийнято рішення обрати мову програмування Java, саме його фреймворк Spring для зручної створення веб-додатка. Головні критерії, на які слід звернути увагу, що даний фреймворк надає додаткові модулі та компоненти для роботи з даними та для захисту розроблючого веб-додатку, а саме : Spring Data, Spring Security, Spring MVC, тощо. Також фреймворк Spring забезпечує принцип Inversion of Control, що допомагає писати слабко-зв’язаний код. Адже згідно вказаним принципом говорить, що відповідальність за створення об’єктів буде лежати на зовнішньому компоненті. Інверсія управління реалізована декількома способами, серед яких є Dependency Injection і пошук залежностей Dependency Lookup. Найчастіше використовують Dependency Injection через конструктор так, як це надає великої гнучкості розроблюючого продукту. Тобто, використовуючи фреймворк Spring звільняє не тільки від необхідності створювати об'єкти, але і зв'язувати їх, що значно полегшує написання програмного коду.

Що до реалізації клієнтської частини був обраний Bootstrap. Дана технологія є дуже легкою для вивчення і це скорочує час на розробку програмного продукту. В даній технології використовуються всі сучасні технічні засоби в області CSS і HTML. Слід відмітити про один з головних плюсів даного фреймворку – це кросплатформність, що в свою чергу його використання доступна на будь-якому браузері в незалежності від операційної системи.

Розділ 3

Реалізація веб-додатку

Для коректної роботи програмного забезпечення необхідно спочатку правильно організувати його структуру та функціонал. Це має великий вплив на швидкість розробки та майбутню підтримку та розширення веб-додатку.

3.1. Вимоги до функціоналу додатку

Неавторизованому користувачеві необхідно пройти напочатку реєстрацію, а потім аутентифікацію і авторизацію, що в свою чергу надає основний функціонал за для використання веб-додатку у своїх цілях.

Неавторизований користувач має можливість:

- Зареєструватися;
- виконати аутентифікацію.

Авторизований користувач має основний функціонал за для користуванням даним веб-додатком:

- Перегляд головної сторінки;
- перегляд усіх дописів інших користувачів;
- можливість ставити уподобання до дописів інших користувачів;
- можливість підписатися або відписатися від користувачів, дописи яких вам сподобалися;
- можливість перейти на профіль певного користувача, який вам до вподоби та переглянути всі його;
- можливість створювати власні дописи;
- можливість додавати до дописів мультимедійні картинки;
- перегляд власних дописів;
- можливість редагування власних дописів;
- зміна особистих даних у вкладці профіль;
- можливість перегляду власних підписок;
- можливість перегляду користувачів, котрі підписані на ваші дописи.

Адміністратор має функціонал подібний до звичайного користувача але з більшими привілегіями:

- можливість додавати нових модераторів;
- можливість змінювати існуючі пости користувачів, якщо ті порушують конфіденційні права додатку

3.2. Опис обраної архітектури веб-додатку

3.2.1. Model View Controller

Для проєктування даного програмного забезпечення був використаний архітектурний шаблон проєктування MVC[21]. Що в свою чергу розбиває структуру додатку на три компоненти, які ми можемо бачити на рис 3.1.

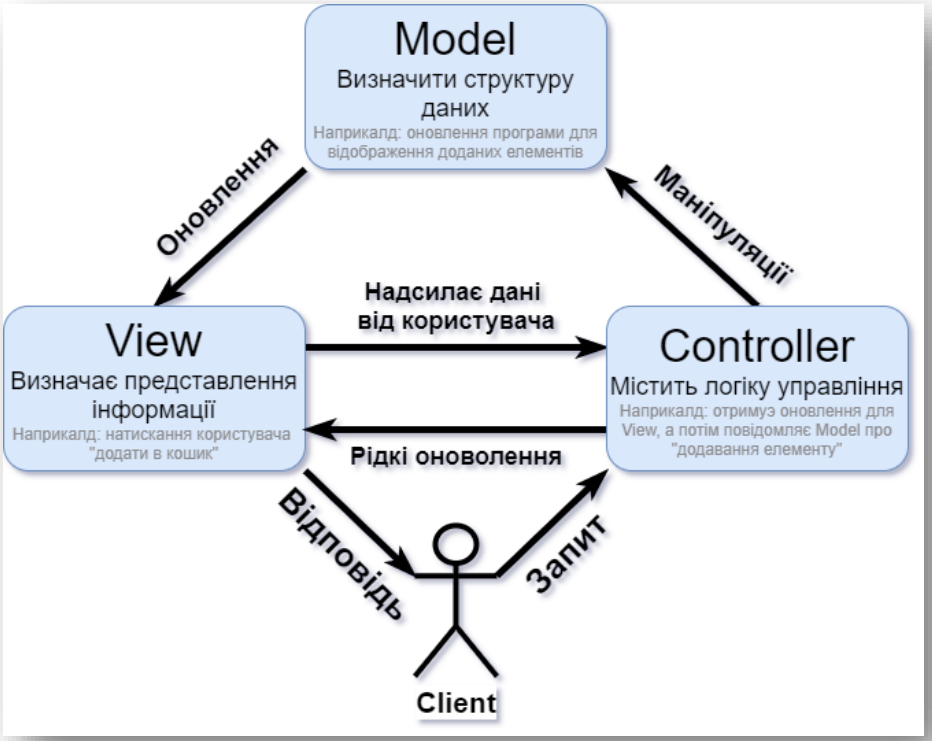


Рис. 3.1. Структура шаблону проєктування MVC

- Model – основний компонент даного шаблону проєктування. Це динамічна структура даних програми яка ніяк не залежить від

інтерфейсу користувача. Особисто він керує усіма даними, логікою та правилами програми.

- View – це компонент архітектури, який відповідає за будь-яке представлення інформації в програмі, наприклад таблиця, графіки чи діаграми.
- Controller – Приймає ввід даних та перетворює його в команди для Model чи View.

Слід відмітити, що даний шаблон проєктування хоч і розбиває додаток на представленні компоненти але крім цього він вказує на тісну взаємодію між даними компонентами[20].

На рис 3.2 зображена структурна схема програмного забезпечення яке ми розробили.

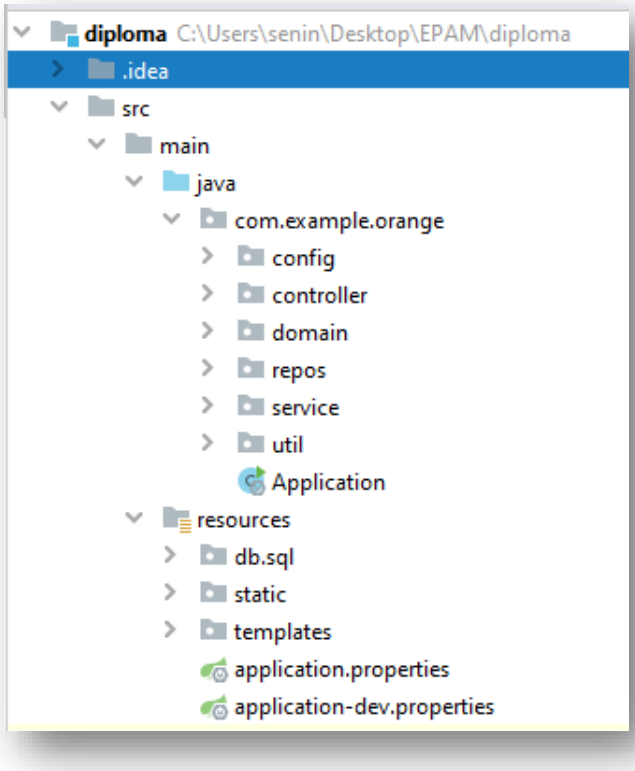


Рис. 3.2. Структура розробленого веб-додатку

Головні компоненти розробленого програмного забезпечення є структурованими і вони зберігаються в окремих директоріях(package):

- config - конфігурації веб-додатку;
- controller - один із компонентів MVC, який отримує дані з View, групує їх в команди і перенаправляє їх на модель;
- domain – елемент модельної частини, який відповідає за роботу з даними програмного забезпечення;
- repos – відповідає за зв'язок програмного забезпечення з базою даних;
- service – це елемент моделі, який відповідає за обробку бізнес логіки і помилок;
- util – це додаткові класи, які виконують проміжну сервісну роботу;
- Application.java – це клас який запускає веб-додаток;
- Resource – це директорія, яка містить файли клієнтської частини програмного забезпечення;
- db.sql – це директорія для міграції з базою даних;
- static – в даній директорії знаходяться файли JavaScripy, Css, etc.;
- templates – знаходять веб сторінки написані на мові HTML;
- application.properties – загальні налаштування для програмного забезпечення.

Нижче наведені основні переваги та недоліки шаблону проєктування MVC[19].

Переваги:

- Одночасний розвиток. Тобто кілька різних моделей можуть одночасно працювати над Model, View та Controller;
- згуртованість – даний шаблон дозволяє логічно групувати зв’язані дії на контролері разом, а також дії для конкретної моделі;

- проста модифікація - через розподіл обов'язків майбутні модифікації, а також розвиток додатку стають зручнішими;
- незалежність - при чіткішому розділенні проблем, є можливість перевіряти самостійно кожну. Наприклад, працюючи з Model можна незалежно виправляти View;
- зрозуміла структура всієї програми;
- легка підтримка програмного забезпечення;.

Недоліки шаблону проєктування MVC:

- незручна для невеликих програм, що негативно впливає на продуктивність та дизайн програми;
- Повинні бути чіткі правила щодо методів;
- використання великої кількісті ресурсів.

3.2.2. Клієнт серверна архітектура

Для створення та розробки даного програмного забезпечення було обрано клієнт-серверну архітектуру.

Дану архітектуру можна описати, як концепцію інформаційної мережі, яка передбачає зосередження головної частини ресурсів в серверах, які в свою чергу обслуговують клієнтів[17]. Слід відмітити, що дана архітектура має такий набір компонентів:

- Набір серверів, які в свою чергу надають інформацію та інші сервіси` програмам, які надсилають певні запити до них;
- набір клієнтів, що користуються серверами, які в свою чергу мають певний набір послух необхідних для користувача;
- мережа, яка безпосередньо забезпечує взаємодію між клієнтом та сервером.

Модель клієнт-серверної архітектури передбачає розділення обов'язків між клієнтом(користувачем) та сервером. Існує три основних рівня операцій:

- рівень представлення даних. Тобто даний рівень являє собою інтерфейс для користувача, який відповідає за представлення даних до клієнта та введення основних команд користувачем;
- прикладний рівень. Він реалізує основну логіку програмного забезпечення і завдяки ньому здійснюється обробка інформації, яку запросив клієнт;
- рівень управління даними. Третій рівень забезпечує додавання, збереження та доступ до даних.

Клієнт може мати багато аспектів, зокрема можна говорити про персональний(клієнтський) комп'ютер, з якого надсилаються звернення до інших комп'ютерів. Також слід сказати про клієнтське та серверне програмне забезпечення. І також слід відзначити людей, які за допомогою певного програмного можуть отримати доступ до бажаної іншої інформації.

Слід зауважити, що у великих додатках типу «Google», «Netflix» кількість клієнтів в рази перевищує кількості серверів але це не впливає на швидкість виконання запитів надіслані клієнтами. На рис 3.3. зображено схему клієнт-серверної архітектури зображено.

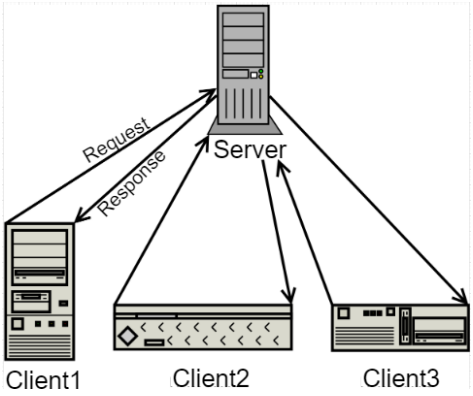


Рис. 3.3. Схема клієнт-серверної архітектури

Отже дана архітектура працює так, клієнти надсилають запити до сервера, який в свою чергу приймає, зберігає, оброблює та повертає дані, які були вказані в запиті. З цього слідую що саме сервер є одним із головних компонентів представленої архітектури. Відповідно до розглянутого слід

сформувати висновок, що сервер хоч і я чистиною розглянутої архітектури і повинен працювати в парі з клієнтом, однак сервер може бети використаний, як окремий і незалежний компонент, що в свою чергу дозволяє працювати без взаємодії з клієнтом. А що до клієнта, в нього така можливість відсутня, так як він не здатний працювати без взаємодії з сервером.

Переваги:

- Усі файли зберігаються в центральному місці
- мережева периферія управляється централізовано;
- резервне копіювання та безпека мережі контролюється централізовано;
- користувачі можуть отримувати доступ до спільних даних, які контролюються централізовано;
- налаштування власного рівня безпеки.

Недоліки:

- Потрібна спеціалізована мережева операційна система;
- сервер дорого придбати;
- потрібен спеціалізований персонал, такий як менеджер мережі;
- Жорстка залежність клієнтської частини від роботи серверної частини, якщо будь-яка частина мережі вийде з ладу, може статися багато порушень.

Отже клієнт-серверна архітектура розподіляє навантаження і роботу по доступним серверам, тим самим надає швидкості і ефективності при роботі з клієнтами. Слід зауважити, що між клієнтською і серверною частиною також чітко розподілена робота, що сприяє високої продуктивності даної архітектури.

3.2.3. Серверна частина

Серверна частина розробленого програмного забезпечення складається з таких компонентів, як Web-Server, Controller, Service, Repository, Model, Service(Entity), база даних.

Web-server надсилає запит в Controller, який оброблює та передає вже оброблені дані на Service. В Service проходить виконання певного алгоритму, бізнес-логіки програм. У разі необхідності будь яких даних Service надає запит до Repository, який в свою чергу звертається до вказаної бази-даних, в нашому випадку це PostgreSQL. Після виконання запиту база-даних повертає дані в Entity який в свою чергу повертає дані в Repository. Слідуючи Repository повертає в Service, а Service в свою чергу оброблює дані згідно з запитом Controller і після виконання цих усіх дій, результат оброблених даних повертається на Controller, який відповідає на запит користувача і представляє достовірні дані користувачеві. На рис 3.3 зображена дана структурна схема, яка використовується в представленому програмному забезпеченні.

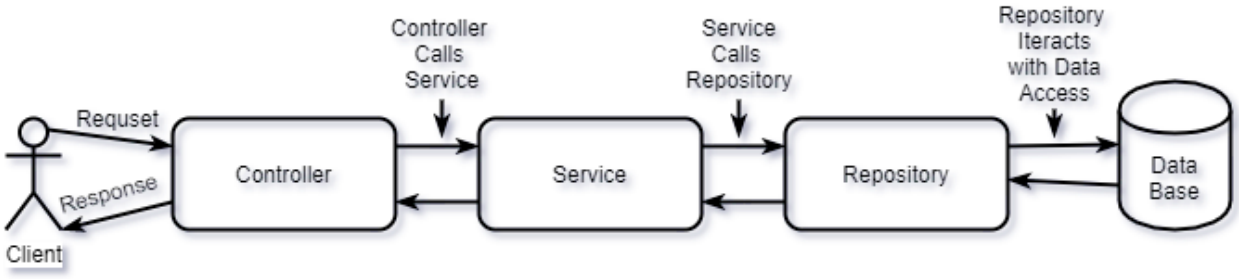


Рис. 3.3. Структурна схема веб-додатку.

Пропоную розглянути запропоновану схему на рис. 3.3.

3.2.3.1. Controller

Як вказано раніше, сам Controller отримує дані від Model або View та перетворює їх на конкретні команди, які будуть передані далі відносно структурної схеми на рис. 3.3.

На рис. 3.4 зображені програмні класи контролера.

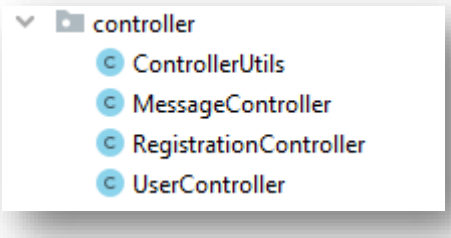


Рис. 3.4 Програмні класи Controller

3.2.3.2. Service

Service – це система, яка обробляє дані надіслані з контролера. Іншими словами це механізм, який дозволяє керувати бізнес логікою. На рис 3.5 зображено директорію з сервісами у розробленому веб-додатку.

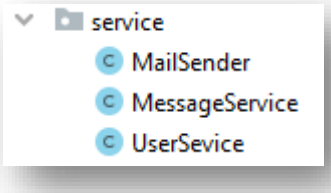


Рис. 3.5. Сервіси програми

3.2.3.3. Repository

Repository- це система, яка за призначена для доступу даних до моделі і за допомогою JDBS-драйвера, репозиторій надсилає запит до бази даних для збереження, редагування, витягування об’єктів чи даних. На Рис. 3.6. зображені програмні класи репозиторію.

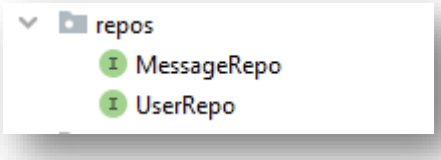


Рис. 3.6. Репозиторії веб-додатку

3.2.3.4. Model

Model являє собою модель з усією бізнес-логікою програми. Дана частина MVC архітектури найбільш незалежна, тому що саме в даному модулі містяться код та об’єкти, які надають сутності програмному забезпеченню, що можуть відповідати реальним об’єктам. На рис. 3.6 зображено вигляд моделі в програмному коді додатку.

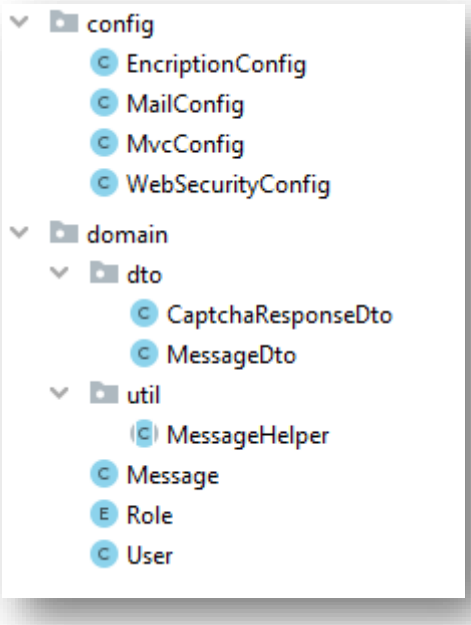


Рис. 3.6. Вигляд моделі в директоріях веб-додатку

3.2.3.5. СУБД PosgreSQL

Для створення розробленого веб-додатку було використано СУБД PosgreSQL, так як в майбутньому планується удосконалення та розширення.

Як було описано вище, щоб використовувати, додавати, змінювати дані з бази даних проходить ряд послідовних дій, початок яких іде від запиту користувача в Controller.

При створенні бази даних було визначено сім таблиць, які можемо бачити на рис. 3.7.

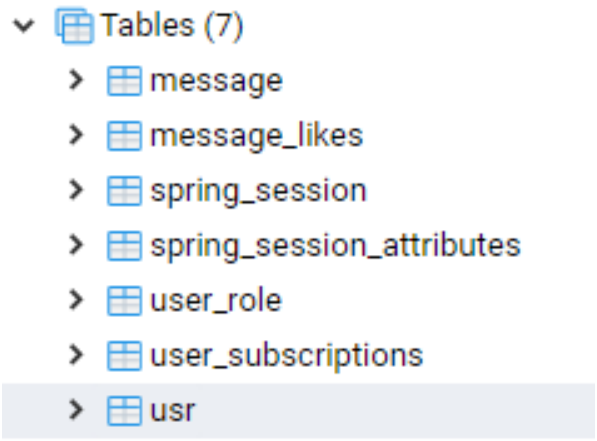


Рис. 3.7. Доступні таблиці веб-додатку

Пропоную розглянути декілька основних таблиць. Перша таблиця - «message», в які зберігаються усі повідомлення, які будуть створенні користувачем. На рис. 3.8. зображено повідомлення, які вже присутні в веб-додатку.

```
1  SELECT * FROM public.message
2
```

Data Output | Explain | Messages | Notifications

	id [PK] bigint	filename character varying (255)	tag character varying (255)	text character varying (2048)	user_id bigint
1	1	a36ebcc5-6c8e-436f-a843-aa...	Greeting	Hallo every one!	1
2	2	c48d3050-fe9f-4b26-9c8f-88e...	WOW	Look at my new desctop img	1
3	3	[null]	let`s go	Subscribe to me and u will see ...	1
4	5	[null]		@admin u have a nice profile	4
5	6	15a441de-f803-4dee-9738-5f...		this is my business schem	4

Рис. 3.8. перегляд вмісту таблиці «message».

З Рис. 3.8. можемо бачити поля які присутні в даній таблиці, а саме:

- «id» – автоінкрементне поле, яке слугує для ідентифікації повідомлень по id;
- «filename» – вказує на ім’я файлу, який може бути завантажено до певного повідомлення у разі його відсутності відповідне поле становить [null];
- «tag» – вказує на тег повідомлення. що допомагає швидко знаходити потрібні нам новини;
- «text» – це безпосередньо вміст самого повідомлення;

- «user_id» – вказує на автора повідомлення, який відображається по id.

Наступна таблиця – «message_likes». Вона необхідна для ідентифікації уподобань, які прив'язані до користувача якому дана новина сподобалась. На Рис. 3.9 зображено повідомлення, які були до вподоби іншим користувачам.

```
1  SELECT * FROM public.message_likes
2
```

Data Output | Explain | Messages | Notifications

	message_id [PK] bigint	user_id [PK] bigint
1	1	1
2	2	4
3	1	4
4	3	4
5	6	4
6	6	7
7	5	7

Рис. 3.8. Перегляд вмісту таблиці «message».

В даній таблиці присутні тільки два поля:

- «message_id» - для ідентифікації повідомлення;
- «user_id» - id користувача, який уподобав конкретне повідомлення.

Розглянемо таблицю «usr», в які зберігаються усі дані користувачів. Слід відмітити, що за для безпеки пароль користувача було зашифровано. На Рис. 3.9. представлені усі зареєстровані користувачі.

```
1  SELECT * FROM public.usr
2
```

Data Output | Explain | Messages | Notifications

	id [PK] bigint	activation_code character varying (255)	active boolean	email character varying (255)	password character varying (255)	username character varying (255)
1	1	[null]	true	[null]	$2a$08$ULxMRzpvRfDw/Mz...	admin
2	4	[null]	true	senin.yura99@gmail.com	$2a$08$Ym4z.ksEPkQKQZaR...	Yurii Sienin
3	7	5c22d795-ee2a-4875-bddd-af...	true	samas85030@provlst.com	$2a$08$wXABwa2fRWpx2Ek...	Pavlo99

Рис. 3.9. Представленні всі зареєстровані користувачі.

Слідуючи з представленої таблиці можемо спостерігати 6 полів:

- «id» ідентифікація користувача;
- «activation_code» - унікальний код активації, який приходить користувачеві на поштову скриньку;
- «active» - стан аккаунту, який виконуються після реєстрації. Існує два стани аккаунту це True і False. Аккаунту надається статус True у разі натискання унікального коду, який надсилається користувачеві на поштову скриньку під час реєстрації;
- «email» - це поштова скринька користувача, яка вказується під час реєстрації;
- «password» - пароль до аккаунту зареєстрованого користувача, який шифрується під час запису до бази даних
- «username» - ім’я зареєстрованого користувача.

ВИСНОВКИ ДО РОЗДІЛУ 3

Протягом третього розділу були розглянуті основні вимоги, та реалізовані запропоновані технології, які були детально описані. Також розглянули СУБД PosgreSQL, яку використовує розроблений веб-додаток. Додаток використовує архітектурний шаблон MVC і клієнт-серверну архітектуру, що значно покращує розробку і удосконалення програмного продукту.

Що до функціональності веб-додатку, то він відрізняється для кожної ролей користувачів. Досліджений функціонал, як для неавторизованих користувачів, так і для авторизованих користувачів двох типів: адміністратора і звичайного користувача.

Реалізований архітектурний шаблон проєктування Model, View, Controller. А також дослідженні переваги та недоліки використання даного стилю розробки..

Описана реалізація клієнт-серверної архітектури, її загальні положення, переваги та недоліки. Детально проаналізовані компоненти серверної частини та наведені приклади використання їх в програмному коді.

Детально описана структура розробленої СУБД PostgreSQL. Представлені та проаналізовані основні таблиці, які використовує веб-додаток.

Для створення запропонованого програмного забезпечення було використано :

- Для серверної частини мову програмування Java з використанням фреймворку Spring;
- для клієнтської частини було вирішено використано фреймворк Bootstrap – для розробки клієнтської частини разом з використанням HTML та CSS;
- для зв’язком з бази даних було використано СУБД PostgreSQL.

РОЗДІЛ 4
ПРЕДСАВЛЕННЯ РОЗРОБЛЕНОГО ВЕБ-ДОДАТКУ

В даному розділі пропонується продемонструвати користувацький інтерфейс та інструкцію, що до користування запропонованого веб-додатку.

4.1. Демонстрація роботи запропонованого програмного продукту

Розроблене програмне забезпечення являє собою веб-додаток. Було спроєктовано зручний та легкий для користування інтерфейс. Слід зауважити, що демонстрація і інструкція з користуванням буде винесено в один розділ, так як ці питання будуть дублювати один одного і описувати одне й теж саме немає необхідності.

Щоб почати користуватися даним веб-додатком, необхідно скомпілювати програмний код в середовищі для розробки IntelliJ IDEA 2019.2.4.(Ultimate Edition), яке в свою чергу має зв'язок з базою даних (PostgreSQL). Слід зауважити, що для запуску програмного коду необхідно встановити JDK 1.8(JavaVersion8). Після успішного компілювання слід відкрити браузер і стрічці пошуку прописати «localhost:9000».

Після виконання ряду дій, які описані вище ми можемо розпочати користуватися представленим веб-додатком. Перейшовши по даному посиланню користувач потрапляє на гостьову сторінку, яка зображена на Рис.4.1. і ви відразу відстежується в системі як неавторизований користувач.

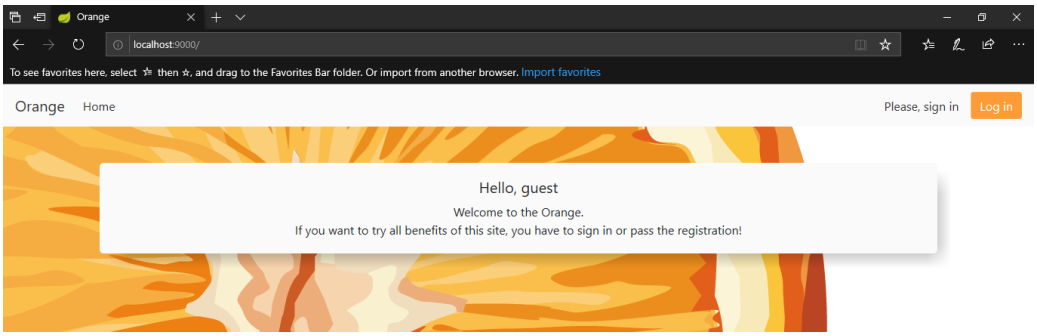


Рис. 4.1. Відображення гостьової сторінки

Щоб зареєструватися не авторизованому користувачеві необхідно натисну лівою кнопкою миші на кнопку «Log in», яка розташована у правому

верхньому кутку веб сторінки. Користувач потрапляє на сторінку авторизації, після чого йому необхідно натиснути на кнопку «Add new user» на сторінку реєстрації. Можемо спостерігати сторінку авторизації на Рис. 4.2.

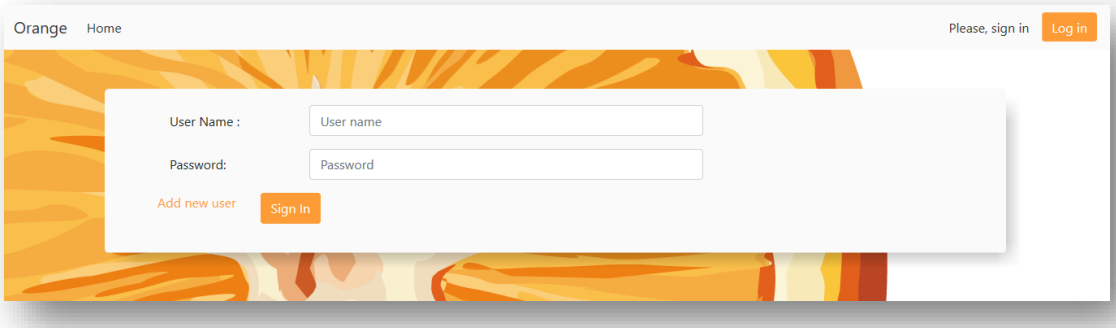


Рис. 4.2. Сторінка авторизації веб-додатку

Перейшовши на сторінку реєстрації користувачеві потрібно заповнити всі позначенні поля, а саме «User Name», «Password», «Repeat Password», «Email» і пройти Captcha перевірку. На Рис. 4.3. зображена сторінка реєстрації.

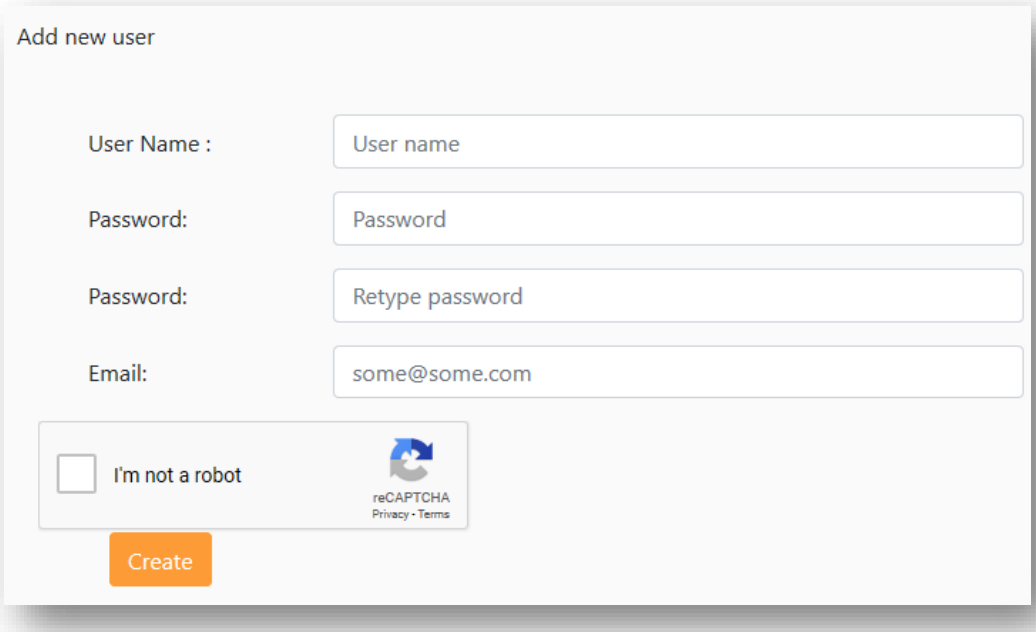


Рис. 4.3. Сторінка реєстрації

У разі заповнення не усіх полів, реєстрації не відбудеться і користувачеві навпроти поля з неправильними або порожніми даними висвітлиться помилка, і буде вказано, які дані через які виникла помилка, ми можемо спостерігати це на Рис. 4.4.

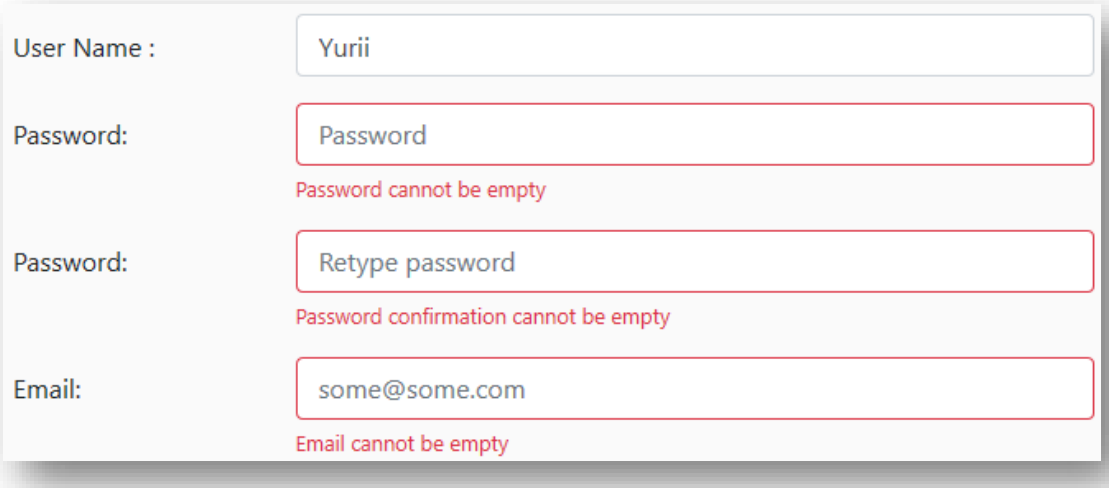


Рис. 4.4. Сторінка реєстрації з невірними даними

Після успішного заповнення усіх полів, ваш аккаунт буде створено але не активовано. Тож для завершення реєстрації користувачеві необхідно перейти на поштову скриньку, яка вказувалася під час реєстрації, на яку було надіслано унікальне запрошення, яке містить лінку веб-додатку. На Рис. 4.5. зображено власне дане запрошення.

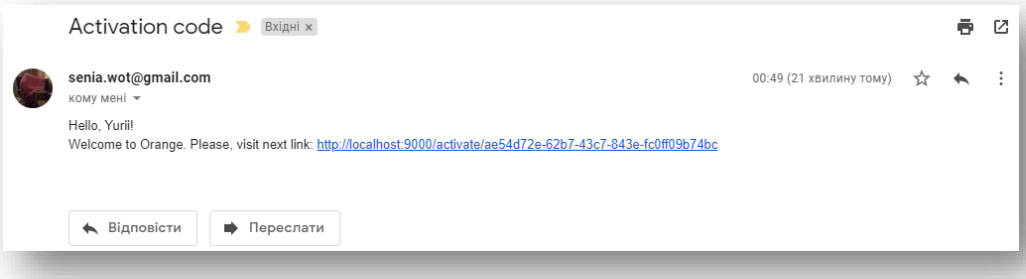


Рис. 4.5. Вигляд запрошення

Після натискання на лінку запрошення, ви будите переадресовані на сторінку для авторизації(вигляд її зображено на Рис. 4.2.), в якій вам потрібно

указати дані: логін і пароль, які ви вказували під час виконання реєстрації, після внесення даних ми натискаємо кнопку «Sign in» і перенаправляємось на гостьову сторінку для авторизованих користувачів.

Так як в даному веб-додатку присутні 2 ролі: звичайний користувач «User», так і головний адміністратор «Admin», тому пропоную розглянути спочатку вигляд функціоналу додатку для звичайного користувача, а потім для адміністратора.

Отже після виконання авторизації та аутентифікації вам надається роль «User», що надає можливість користування запропонованим веб-додатком. У верхній частині екрану у вас з'являється навігаційне меню, яке зображене на Рис. 4.6. і буде нижче описане.

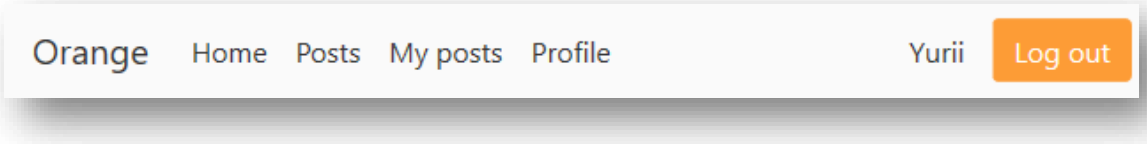


Рис. 4.6. Навігаційне меню

В меню навігації доступні вкладки «Home», «Post», «My Post», «Profile» і кнопка «Log out».

«Home» - це головна сторінка, в якій в майбутньому буде вказано інформації про сайт.

На сторінці «Post», яка зображена на Рис. 4.7. відображаються усі повідомлення, котрі були залишені будь-яким авторизованим користувачем.

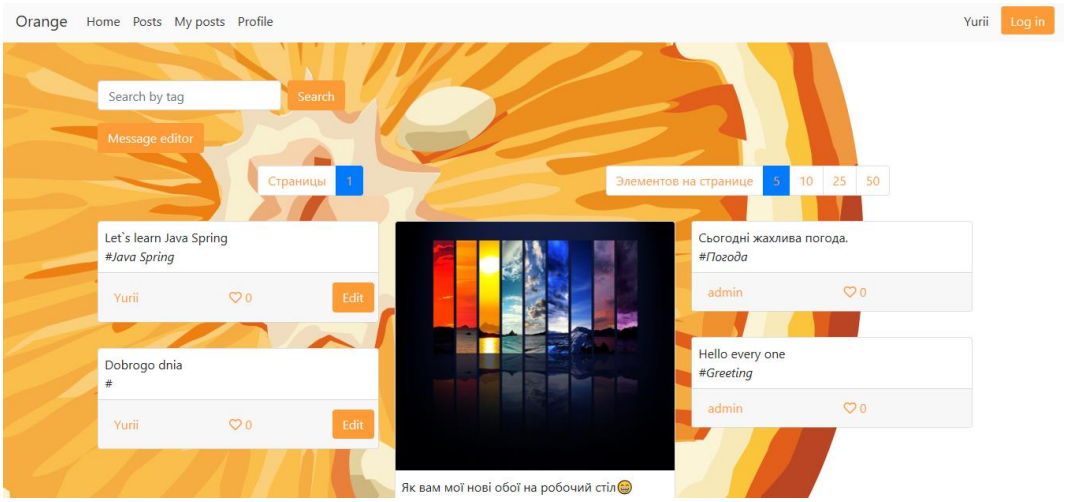


Рис. 4.7 Вигляд веб сторінки «Post»

Перш за все слід відмітити можливість пошуку повідомлень по тегу, що дуже зручно, якщо вам потрібно знайти інформацію на конкретну тему. Щоб створити пост користувачеві необхідно натиснути кнопку «Message editor» і заповнити поля: «Input your message», «Tag» і якщо ви хочете додати зображення да вашого повідомлення потрібно вибрати файл натиснувши «Choose file» або «Browse». Слід зауважити що пост без повідомлення не буде сприйматися системою. Вигляд «Message editor» буде зображено на Рис. 4.8.

Рис. 4.8 Вигляд вкладки додавання повідомлення

Для зручності користування було створено можливість вибору кількості повідомлень, які будуть відображені на одній сторінці, наприклад користувач вибере 5 повідомлень, то наступні будуть надіслані на наступну сторінку. Можемо побачити це на Рис. 4.7.

Перейдемо до самих постів(повідомлень). Після того, як користувач додав новий пост він автоматично переходить на початок сторінки, а наступні

повідомлення згруповано і лаконічно перелаштовуються вперед. Прикладом слугує зображення на Рис. 4.9

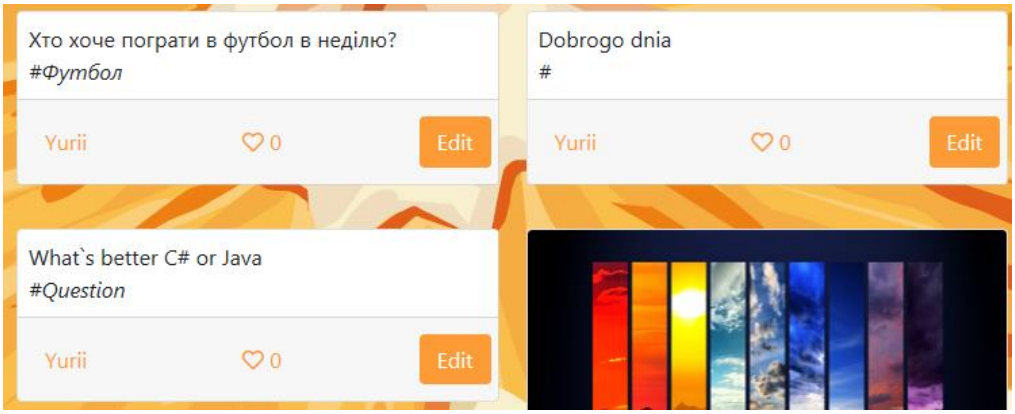


Рис. 4.9 Вигляд повідомлень

Слідуючи з Рис. 4.9. бачимо що кожен пост містить в собі: власне саме повідомлення, певний тег(у разі його відсутності символ «#»), автора, кількість уподобань, і якщо ви створили дане повідомлення, то присутня можливість його редагування. У разі зацікавленості певного користувача, ми можемо натиснути на його ім’я та перейти на його сторінку, де ми можемо спостерігати усі його пости.

Перейдемо до наступної вкладки навігаційного меню, а саме «My Posts». В даному розділі програми відображаються власні пости користувача, які він може переглядати, додавати і редагувати, а також меню підписників і підписок, яке користувач може переглядати. На Рис. 4.10. зображено частину функціоналу даної сторінки, а саме форму підписок і підписників адже весь інший функціонал описано вище.

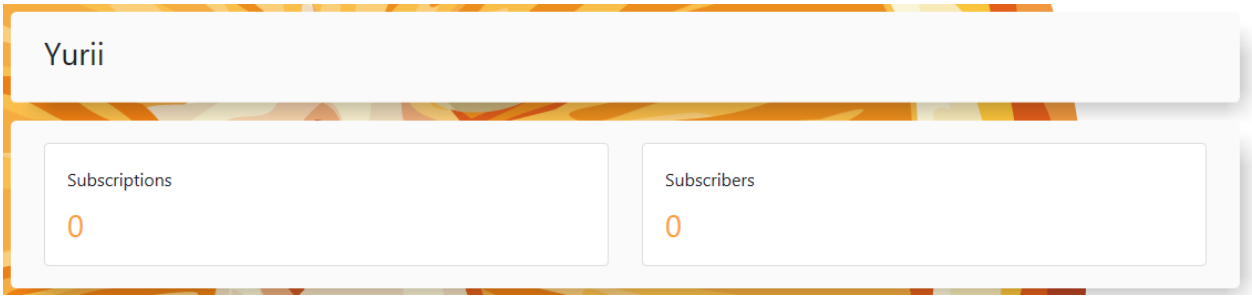


Рис. 4.10 Вигляд сторінки «My posts»

Перейдемо до останньої сторінки «Profile». На даній сторінці користувач може змінювати свої особисті дані а саме: поштову скриньку та пароль. Як і меню «My Posts» присутня форма з вашим іменем і та меню підписок/підписників. На Рис. 4.11. зображено вигляд сторінки «My posts»

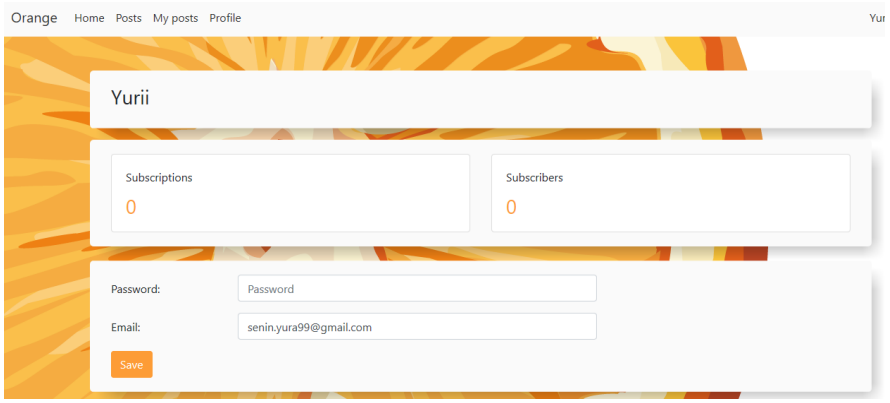


Рис. 4.10 Вигляд сторінки «My posts»

Перейдемо до ролі адміністратора. Після першого запуску веб-додатку в системі присутній адміністратор по замовчанням, у якого у разі необхідності є можливість додавання нових адміністраторі-модераторів. Вдалим рішенням було приховати роль адміністратора відносно інших користувачів. В цілому адміністратор має весь функціонал звичайного користувача але з привілегіями. Користувач в який використовує опубліковуй запис адміністратора слідкує за усіма постами, які будуть надіслані звичайними користувачами, щоб були виконання усі норми правил сайту. У разі порушень даних правил адміністратор може змінити чи видалити повний пост.

Також змінилася і навігаційна панель, додалася нова вкладка «User List», яка містить усіх користувачів та їх ролі. Як було зазначено вище, у адміністратора є можливість додати нових адміністраторів в яких будуть ті ж самі привілегії, що і головного адміністратора. На Рис. 4.11. зображену оновлену навігаційну панель для адміністратора.

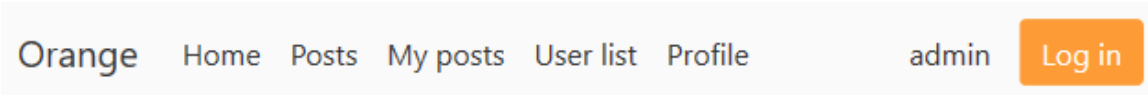


Рис. 4.11 Вигляд навігаційної панель для адміністратора

Що до вкладки «User List», то ми можемо спостерігати на Рис. 4.12.

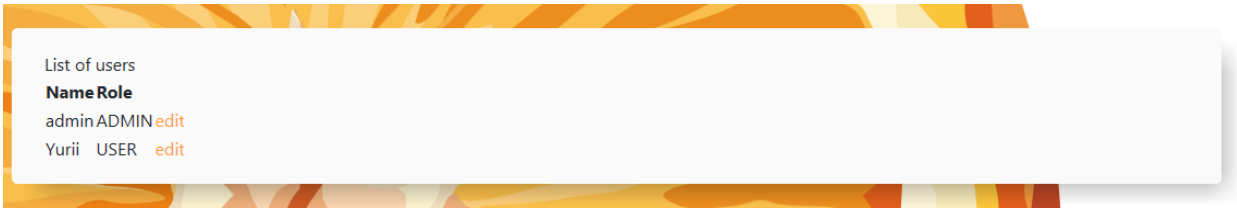


Рис. 4.12. Сторінка «User List»

Слід відмітити, що дане програмне забезпечення пристосоване до різних девайсів навіть мобільних телефонів. Приклад роботи на мобільному телефоні буде наведено на Рис. 4.13. Можна помітити, що навігаційна панель змінилася і тепер, щоб її викликати необхідно натиснути на нову кнопку в правому верхньому вуглі після чого випливе нова панель з навігацією.

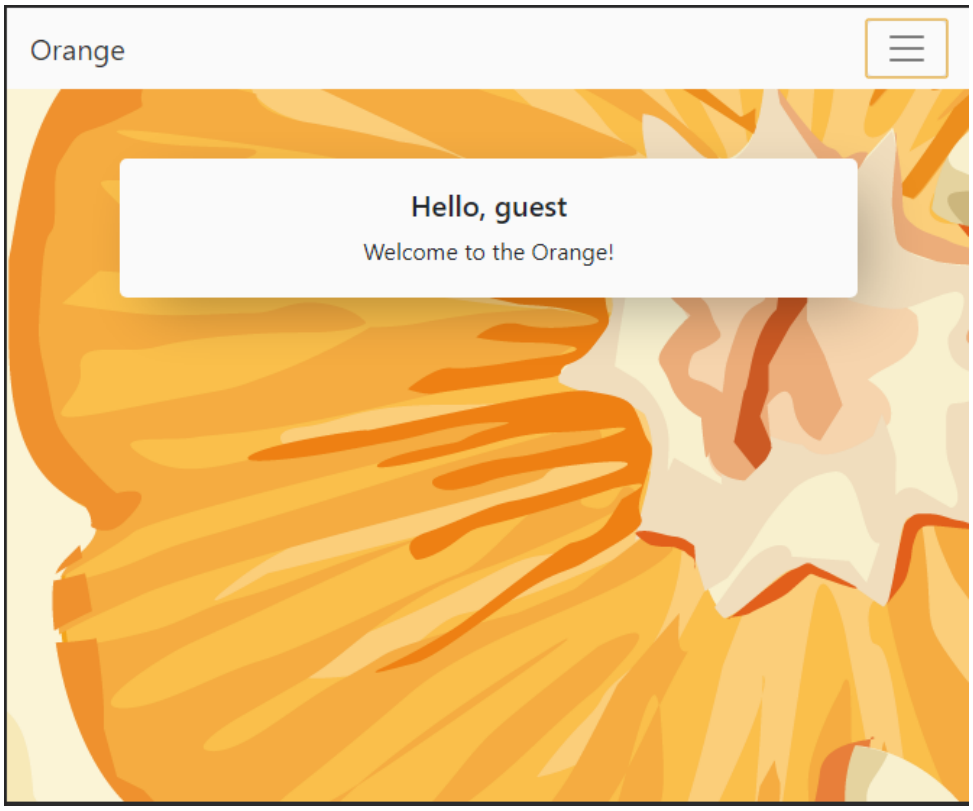


Рис. 4.13. Веб-додаток для мобільних телефонів

На наступному зображені можемо бачити, як змінився інтерфейс додатка на мобільному пристрої Рис.4.14.

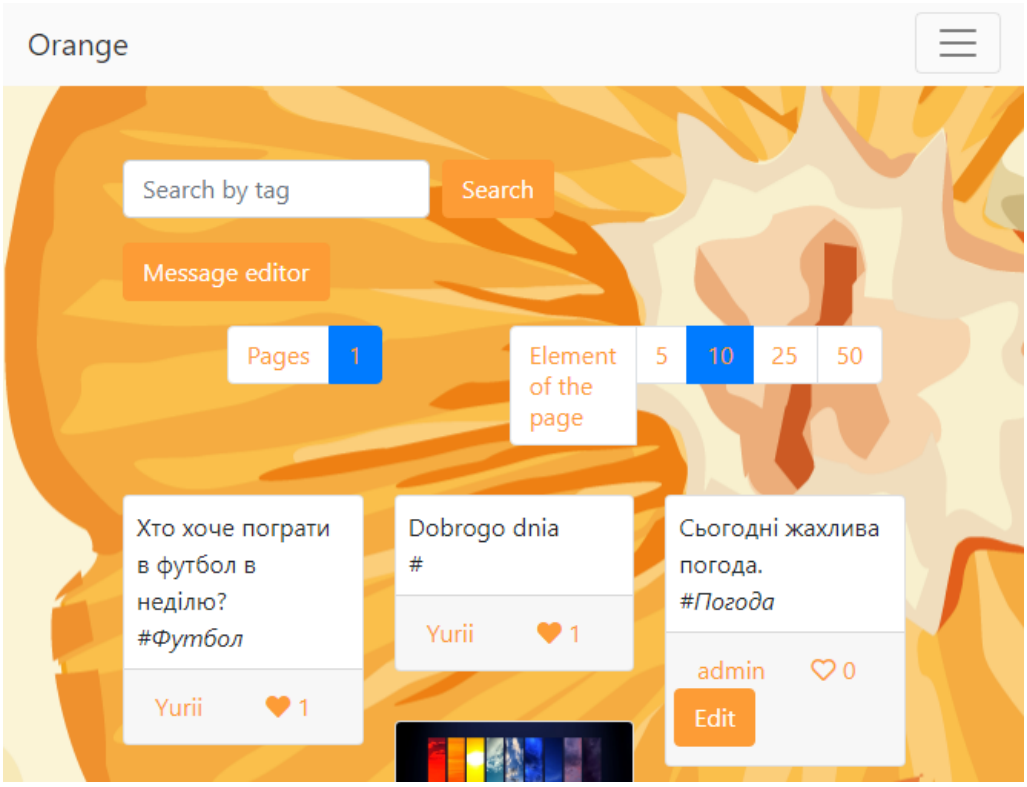


Рис. 4.14. Веб сторінка «Posts» для мобільних пристроїв

4.2. Тестування запропонованого web-додатку

Було розроблено ряд тест-кейсів, які зображені в Таблиці 1 для проведення тестування над функціоналом розробленого програмного забезпечення.

Сценарій до димового тестування:

1. Скомпілювати проєкт.
2. В браузері перейти на посилання «localhost:9000».
3. Перейти на сторінку реєстрації.
4. Ввести дані для реєстрації.
5. Ввійти в систему за допомогою даних, які би вказані при виконанні реєстрації.
6. Додати новий пост.
7. Змінити доданий пост.
8. Поставити уподобання.
9. Підписатися на користувача.

Таблиця 1. - Тест-кейси димового тестування

№ п/п	Опис	Очікувані результати
1	1. Скомпілювати програмний код.	1. Система запущена без помилок і готова до використання через веб браузери любих типів.
2	1. В браузері перейти на посилання «localhost:9000» відображення гостьової сторінки.	1. Система переходить на гостьову сторінку.
3	1. Перейти на вкладку реєстрації.	1. Система переходить та відображає сторінку реєстрації.
4	1. Ввести дані для реєстрації. 2. Користувач вносить дані.	1. Користувач вводить коректні дані, після чого йому на поштову скриньку надходить запрошення з посиланням для активації аккаунту.
5	1. Перейти на сторінку авторизації 2. Введення даних, котрі були вказані під час виконання реєстрації	1. Система переходить на сторінку авторизації. 2. Користувач вносить дані і авторизується на сайті.
6	1. Перейти у вкладку «Post». 2. Додати нове повідомлення з зображенням. 3. Завантажити зображення	1. Система переходить у запропоновану вкладку і відображає сторінку з усіма постами. 2. Користувач додає власний пост з зображенням але без тегу, який відображається на початку даної сторінки. 3.Зображеня завантажено.

Продовження таблиці 1. - Тест-кейси димового тестування		
7	Змінити доданий пост, а саме додати інший тег.	1. Користувач додає тег до свого повідомлення.
8	Поставити уподобання під постом іншого користувача.	1. Після натиску кнопки уподобання, з’являється відповідний символ на пості.
9	Підписатися на користувача.	1. Користувач переходить на сторінку іншого користувача. Після натиску на кнопку «Subscribe», біля імені користувача відображається, що ви на нього підписані.
7	Змінити доданий пост, а саме додати інший тег.	1. Користувач додає тег до свого повідомлення.
8	Поставити уподобання під постом іншого користувача.	1. Після натиску кнопки уподобання, з’являється відповідний символ на пості.

4.3.Рекомендації щодо розвитку і вдосконалення

Для подальшого розвитку даного програмного забезпечення потрібно додати можливість повного редагування власного аккаунту, а також пришвидшити роботу завантажувальних веб-сторінок. Потрібно допрацювати дизайн сайту використовуючи JavaScript, CSS, та інше. Покращити взаємодію між користувачами. Наприклад додати можливість користувачеві помічати інших користувачів, яким в свою чергу надходять повідомлення, що вони були відмічені. Необхідно використовувати систему контролю версій типу «GitHub» або «GitLab», що в свою чергу забезпечить стабільність роботи програмного забезпечення.

ВИСНОВОКИ ДО РОЗДІЛУ 4

Протягом 4 розділу представлено розроблений веб-додаток, який був розроблений протягом виконання дипломної роботи. Були розглянуті питання функціоналу і інструкції для користуванням даним програмним забезпеченням. Детально досліджена робота веб-додатку для усіх користувачів і представлено інтерфейс для кожного з ролей окрім не авторизованого користувача. Наведено приклад роботи в декількох браузерах, а саме це «Google Chrome» і «Microsoft Edge», а також був запущений емулятор для мобільного пристрою, де також була продемонстрована коректна робота даного веб-додатку.

Було виконано ряд тестів димного тестування, які варто відмітити пройшли успішно. Написані тест-кейси, які допоможуть коректно протестувати і в разі необхідності виправити помилки, які можуть виникнути під час проведення даних тестів.

Також слід відмітити, що були написані рекомендації, що до подальшого розвитку програмного забезпечення. Наведено приклади, що варто зробити за для зручного і продуктивного використання.

ВИСНОВКИ

Метою написання запропонованого дипломного проєкту було розробити програмне забезпечення – веб-додаток для зручного користуванням та створення власних мокроблогів.

В першому розділі були розглянуті подібні веб-аналоги, які були детально проаналізовані і досліджені усі їхні переваги та недоліки, що дозволило визначити вимоги до розробленого програмного продукту.

Протягом другого розділу був розглянутий стек технологій для розробки запропонованого програмного продукту. І після детального аналізу було вирішено, що краще слід використовувати для зручного користування, захищеності і швидкості веб-додатка.

Серверна частина, архітектурний шаблон, архітектура програми та безпосередньо база даних були розглянуті в третьому розділі. Кожен з представлених пунктів був обраний і детально досліджений і описаний.

Було продемонстровано вже розроблений продукт та представлена детальна інструкція що до користування даним веб-додатком.

Було вирішено обрати мову програмування Java, так як вона універсальна і зручна. Слід зауважити, що особливу увагу в ході розробки представленого програмного продукту було приділено до інтерфейсу користувача, а також функційній частині. Під час розробки були враховані всі вимоги. Які були зазначені в плані, тестування програмного продукту виконано відповідно до усіх затверджених норм та методику тестування програми.

Щодо використання представленого веб-додатку, то він дасть змогу швидко і інформативно доносити інформацію в маси, а також може слугувати специфікованим веб-додатком для внутрішнього менеджменту компанії.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Facebook.com - SEO Wiki [Електронний ресурс]. Режим доступу: https://wiki.rookee.ru/facebook/ (дата звернення 15.04.2020)
2. Facebook.com – Active Traffic [Електронний ресурс]. Режим доступу: https://www.activetraffic.ru/wiki/facebook/ (дата звернення 16.04.2020)
3. Youtube.com – Wikipedia [Електронний ресурс]. Режим доступу: https://en.wikipedia.org/wiki/YouTube (дата звернення 18.04.2020)
4. Переваги та мінуси веб-сервісу Youtube.com – SnobMonkey [Електронний ресурс]. Режим доступу: https://www.snobmonkey.com/the-advantages-and-disadvantages-of-youtube/ (дата звернення 19.04.2020)
5. Twitter - ESRC [Електронний ресурс]. Режим доступу: https://esrc.ukri.org/research/impact-toolkit/social-media/twitter/what-is-twitter/ (дата звернення 20.04.2020)
6. Переваги та мінуси веб-сервісу Twitter- SocialMediaManager [Електронний ресурс]. Режим доступу: https://socialmediamanager.tweetinggoddess.com/2018/04/20/advantages-and-disadvantages-of-twitter/ (дата звернення 22.04.2020)
7. Phyton for web-app [Електронний ресурс]. Режим доступу: https://realpython.com/python-web-applications/ (дата звернення 26.04.2020)
8. Java version 8 Oracle [Електронний ресурс]. Режим доступу: https://www.java.com/ (дата звернення 27.04.2020)
9. The Spring Framework arhitecture [Електронний ресурс]. Режим доступу: https://spring.io/ (дата звернення 27.04.2020)
10. C# - Wikipedia [Електронний ресурс]. Режим https://en.wikipedia.org/wiki/Comparison_of_C_Sharp_and_Visual_Basic_.NET (дата звернення 28.04.2020)
11. Порівняння СУБД MySQL і PostgreSQL— Порівняння MySQL між PostgreSQL [Електронний ресурс]. Режим доступу https://losst.ru/sravnenie-mysql-i-postgresql (дата звернення 30.04.2020)
12. PostgreSQL - об'єктно-реляційна СУБД[Електронний ресурс]. Режим доступу: https://knowledge.allbest.ru/programming/3c0b65625b2bc69b5d53b89421206c27_0.html (дата звернення 29.04.2019)
13. MySQL – СУБД MySQL[Електронний ресурс]. Режим доступу http://strong-planet.com.ua/database.html/ (дата звернення 29.04.2020)

14. Bootstap — основна характеристика SitesBay[Електронний ресурс]. Режим доступу: https://www.sitesbay.com/bootstrap/bootstrap-features-of-bootstrap (дата звернення 03.05.2020)
15. Bootstap: особливості, переваги та недоліки [Електронний ресурс]. Режим доступу: https://dzone.com/articles/bootstrap-and-its-features (дата звернення 03.05.2020)
16. JavaScript [Електронний ресурс]. Режим доступу: https://habr.com/ru/company/ruvds/blog/476286/ (дата звернення 3.05.2020)
17. MVC in Java: [Електронний ресурс]. Режим доступу: https://docs.spring.io/spring/docs/3.2.x/spring-framework-reference/html/mvc.html (дата звернення 12.05.2020)
18. What is the role of Model in MVC? [Електронний ресурс]. Режим доступу: https://stackoverflow.com/questions/47884943/what-is-the-role-of-controller-in-mvc-model (дата звернення 12.05.2020)
19. Web MVC framework: [Електронний ресурс]. Режим доступу: https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm(дата звернення 12.05.2020)
20. Серверна частина веб-додатку [Електронний ресурс]. Режим доступу: http://sites.znu.edu.ua/webprog/lect/1191.ukr.html (дата звернення 13.05.2020)
21. Клієнт-серверна архітектура [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Клієнт-серверна_архітектура/ (дата звернення 10.05.2020)

ДОДАТОК 1

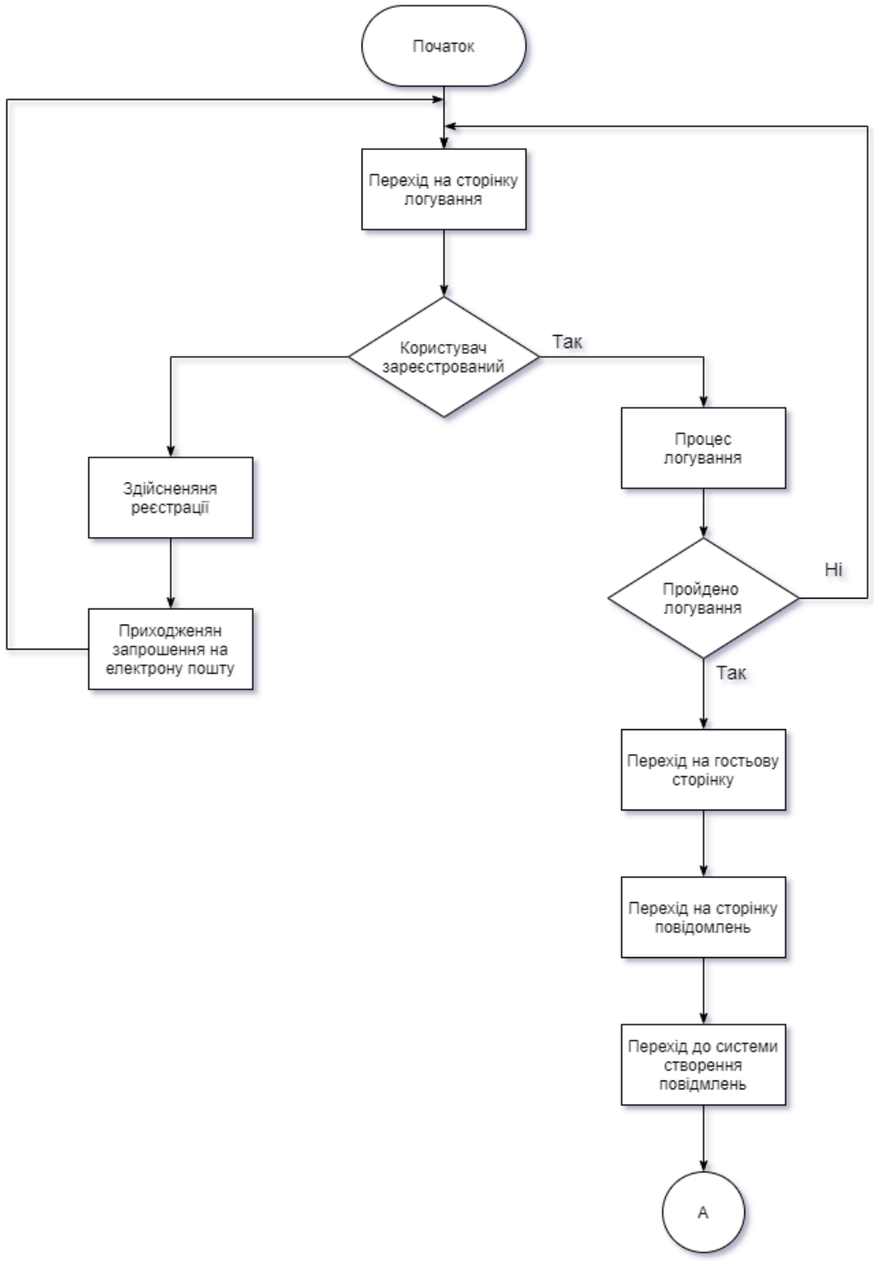
Веб додаток моделі загального публікаційного доступу

Алгоритм дій програмного забезпечення

ІАЛЦ.467100.004 Д1

Аркушів 1

Київ 2020 р



					ІАЛЦ.467100.004 Д1		
Зм.	Арк.	№ докум.	Підпис	Дата	*Веб-додаток загального публікаційного доступу Алгоритм дій програмного забезпечення*	Літ.	Маса / Масштаб
Розроб.		Сенін Ю. І.					
Перевір.		Аленін О. І.				Арк.	Аркушів
Н. контр.		Сімоненко В.П.			*Дипломна робота*	*КПІ ФІОТ кафедра ОТ гр. ІП-64*	
Затверд.							

ДОДАТОК 2

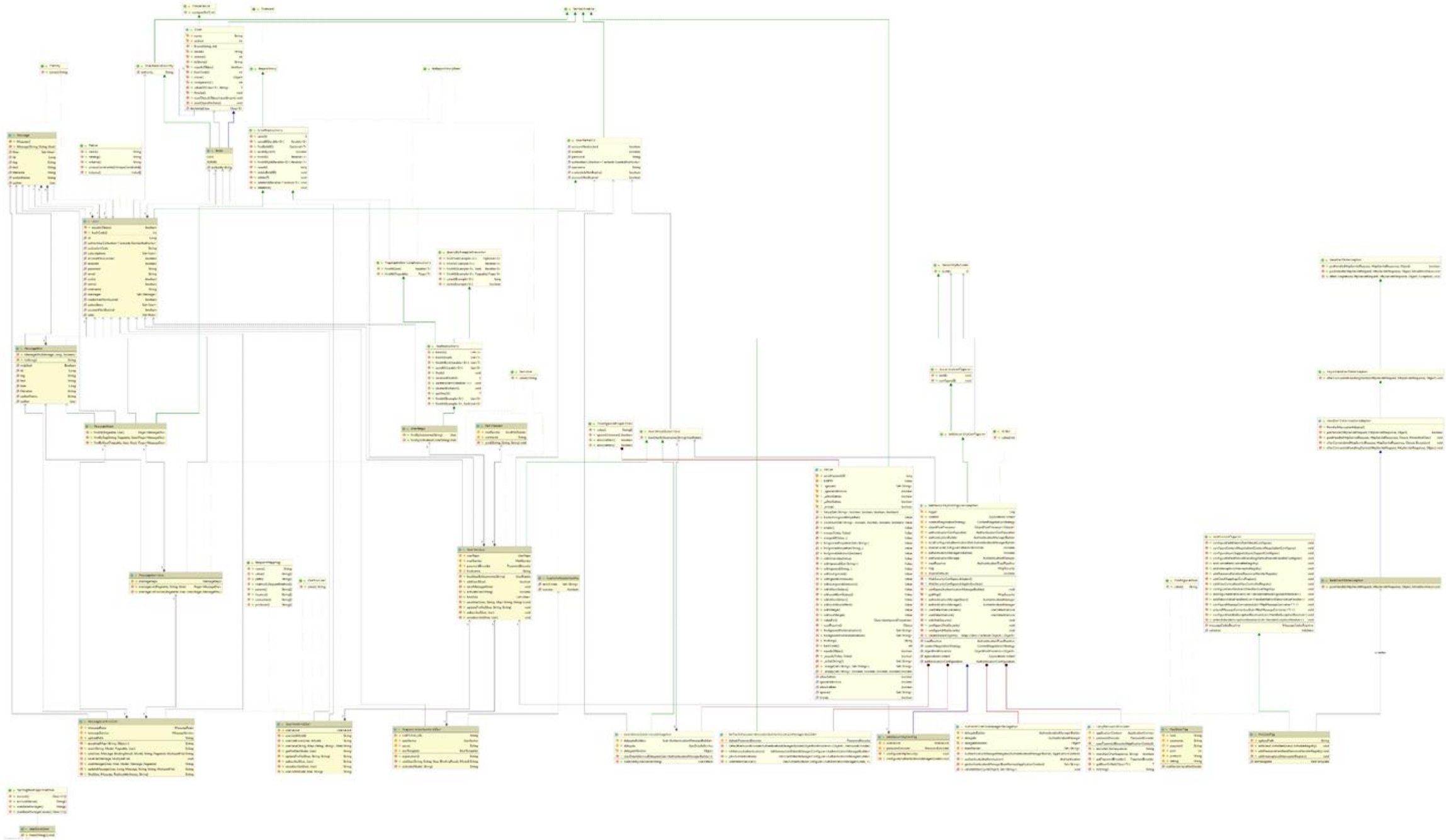
Веб додаток моделі загального публікаційного доступу

Функціональна схема (діаграма класів)

ІАЛЦ.467100.005 Д2

Аркушів 1

Київ 2020 р



						ІАЛЦ.467100.005 Д2	
Зм.	Арк.	№ докум.	Підпис	Дата	*Веб-додаток загального публікаційного доступу Функціональна схема (діаграма класів)*	Літ.	Маса / Масштаб
Розроб.		Сєнін Ю. І.					
Перевір.		Аленін О. І.				Арк.	Аркушів
Н. контр.		Сімоненко В.П.			*Дипломна робота*	*КПІ ФІОТ кафедра ОТ гр. ІП-64*	
Затверд.							

ДОДАТОК З

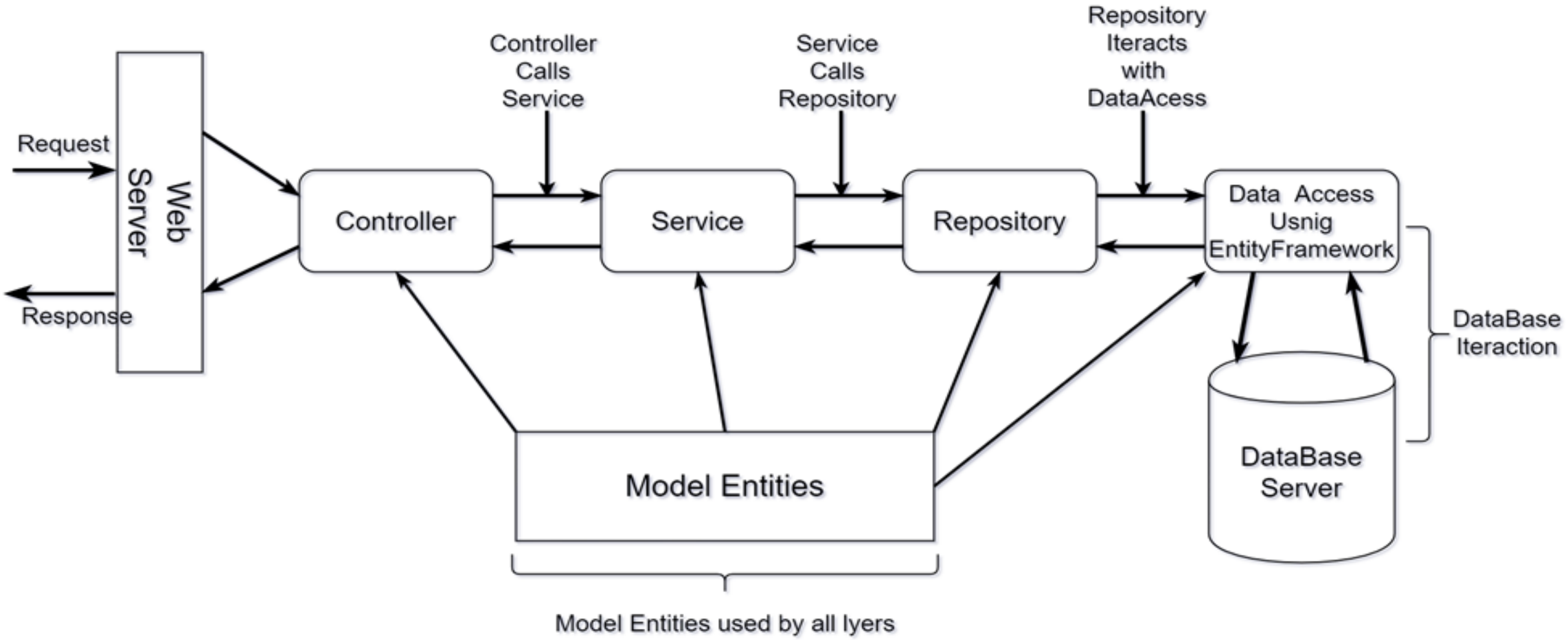
Веб додаток моделі загального публікаційного доступу

Структурна схема веб-застосунку

ІАЛЦ.467100.006 Д3

Аркушів 1

Київ 2020 р



					ІАЛЦ.467100.006 ДЗ		
Зм.	Арк.	№ докум.	Підпис	Дата	*Веб-додаток загального публікаційного доступу Структурна схема веб-застосунку*	Літ.	Маса / Масштаб
Розроб.		Сенін Ю. І.					
Перевір.		Аленін О. І.				Арк.	Аркушів
Н. контр.		Сімоненко В.П.			*Дипломна робота*	*КПІ ФІОТ кафедра ОТ гр. ІП-64*	
Затверд.							

ДОДАТОК 4

Веб додаток моделі загального публікаційного доступу

Текст програми

ІАЛЦ.467100.007.Д4

Аркушів 13

Київ – 2020 року

```java
package com.example.orange.domain;import
org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import javax.persistence.*;
import javax.validation.constraints.Email;
import javax.validation.constraints.NotBlank;
import java.util.Collection;
import java.util.HashSet;
import java.util.Objects;
import java.util.Set;

@Entity
@Table(name = "usr")
public class    User implements UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    @NotBlank(message = "Username cannot be empty")
    private String username;
    @NotBlank(message = "Password cannot be empty")
    private String password;
    private boolean active;

    @Email(message = "Email is not correct")
    @NotBlank(message = "Email cannot be empty")
    private String email;
    private String activationCode;

    @ElementCollection(targetClass = Role.class, fetch = FetchType.EAGER)
    @CollectionTable(name = "user_role", joinColumns = @JoinColumn(name = "user_id"))
    @Enumerated(EnumType.STRING)
    private Set<Role> roles;

    @OneToMany(mappedBy = "author", cascade = CascadeType.ALL, fetch =
FetchType.LAZY)
    private Set<Message> messages;

    @ManyToMany
    @JoinTable(
            name = "user_subscriptions",
            joinColumns = { @JoinColumn(name = "channel_id") },
            inverseJoinColumns = { @JoinColumn(name = "subscriber_id") }
    )
    private Set<User> subscribers = new HashSet<>();

    @ManyToMany
    @JoinTable(
            name = "user_subscriptions",
            joinColumns = { @JoinColumn(name = "subscriber_id") },
            inverseJoinColumns = { @JoinColumn(name = "channel_id") }
    )
    private Set<User> subscriptions = new HashSet<>();

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
```

```
        User user = (User) o;
        return Objects.equals(id, user.id);
    }

    @Override
    public int hashCode() {

        return Objects.hash(id);
    }

    public boolean isAdmin() {
        return roles.contains(Role.ADMIN);
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }

    @Override
    public boolean isAccountNonExpired() {
        return true;
    }

    @Override
    public boolean isAccountNonLocked() {
        return true;
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }

    @Override
    public boolean isEnabled() {
        return isActive();
    }

    public void setUsername(String username) {
        this.username = username;
    }

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return getRoles();
    }

    public String getPassword() {
        return password;
    }
```

```
public void setPassword(String password) {
    this.password = password;
}

public boolean isActive() {
    return active;
}

public void setActive(boolean active) {
    this.active = active;
}

public Set<Role> getRoles() {
    return roles;
}

public void setRoles(Set<Role> roles) {
    this.roles = roles;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getActivationCode() {
    return activationCode;
}

public void setActivationCode(String activationCode) {
    this.activationCode = activationCode;
}

public Set<Message> getMessages() {
    return messages;
}

public void setMessages(Set<Message> messages) {
    this.messages = messages;
}

public Set<User> getSubscribers() {
    return subscribers;
}

public void setSubscribers(Set<User> subscribers) {
    this.subscribers = subscribers;
}

public Set<User> getSubscriptions() {
    return subscriptions;
}

public void setSubscriptions(Set<User> subscriptions) {
```

```
        this.subscriptions = subscriptions;
    }
}
```

```
package com.example.orange.controller;

import com.example.orange.domain.Role;
import com.example.orange.domain.User;
import com.example.orange.service.UserSevice;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.security.core.annotation.AuthenticationPrincipal;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;

import java.util.Map;

@Controller
@RequestMapping("/user")
public class UserController {
    @Autowired
    private UserSevice userSevice;

    @PreAuthorize("hasAuthority('ADMIN')")
    @GetMapping
    public String userList(Model model) {
        model.addAttribute("users", userSevice.findAll());

        return "userList";
    }

    @PreAuthorize("hasAuthority('ADMIN')")
    @GetMapping("{user}")
    public String userEditForm(@PathVariable User user, Model model) {
        model.addAttribute("user", user);
        model.addAttribute("roles", Role.values());

        return "userEdit";
    }

    @PreAuthorize("hasAuthority('ADMIN')")
    @PostMapping
    public String userSave(
            @RequestParam String username,
            @RequestParam Map<String, String> form,
            @RequestParam("userId") User user
    ) {
        userSevice.saveUser(user, username, form);

        return "redirect:/user";
    }

    @GetMapping("profile")
    public String getProfile(Model model, @AuthenticationPrincipal User user) {
        model.addAttribute("username", user.getUsername());
        model.addAttribute("email", user.getEmail());
```

```
        user = (User)userSevice.loadUserByUsername(user.getUsername());

        model.addAttribute("userChannel", user);
        model.addAttribute("subscriptionsCount", user.getSubscriptions().size());
        model.addAttribute("subscribersCount", user.getSubscribers().size());
        model.addAttribute("isSubscriber", user.getSubscribers().contains(user));
        model.addAttribute("isCurrentUser", user.equals(user));
        return "profile";
    }

    @PostMapping("profile")
    public String updateProfile(
            @AuthenticationPrincipal User user,
            @RequestParam String password,
            @RequestParam String email
    ) {
        userSevice.updateProfile(user, password, email);

        return "redirect:/user/profile";
    }

    @GetMapping("subscribe/{user}")
    public String subscribe(
            @AuthenticationPrincipal User currentUser,
            @PathVariable User user
    ) {
        userSevice.subscribe(currentUser, user);

        return "redirect:/user-messages/" + user.getId();
    }

    @GetMapping("unsubscribe/{user}")
    public String unsubscribe(
            @AuthenticationPrincipal User currentUser,
            @PathVariable User user
    ) {
        userSevice.unsubscribe(currentUser, user);

        return "redirect:/user-messages/" + user.getId();
    }

    @GetMapping("{type}/{user}/list")
    public String userList(
            Model model,
            @PathVariable User user,
            @PathVariable String type
    ) {
        model.addAttribute("userChannel", user);
        model.addAttribute("type", type);

        if ("subscriptions".equals(type)) {
            model.addAttribute("users", user.getSubscriptions());
        } else {
            model.addAttribute("users", user.getSubscribers());
        }

        return "subscriptions";
```

```
    }
}
```

```
package com.example.orange.controller;

import com.example.orange.domain.Message;
import com.example.orange.domain.User;
import com.example.orange.domain.dto.MessageDto;
import com.example.orange.repos.MessageRepo;
import com.example.orange.service.MessageService;
import
org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.data.web.PageableDefault;
import
org.springframework.security.core.annotation.AuthenticationP
rincipal;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.util.StringUtils;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;
import
org.springframework.web.servlet.mvc.support.RedirectAttribut
es;
import org.springframework.web.util.UriComponents;
import org.springframework.web.util.UriComponentsBuilder;

import javax.validation.Valid;
import java.io.File;
import java.io.IOException;
import java.util.Map;
import java.util.Set;
import java.util.UUID;

@Controller
public class MessageController {
    @Autowired
```

```
    private MessageRepo messageRepo;

    @Autowired
    private MessageService messageService;

    @Value("${upload.path}")
    private String uploadPath;

    @GetMapping("/")
    public String greeting(Map<String, Object> model) {
        return "greeting";
    }

    @GetMapping("/main")
    public String main(
            @RequestParam(required = false, defaultValue =
"") String filter,
            Model model,
            @PageableDefault(sort = { "id" }, direction =
Sort.Direction.DESC) Pageable pageable,
            @AuthenticationPrincipal User user
    ) {
        Page<MessageDto> page =
messageService.messageList(pageable, filter, user);

        model.addAttribute("page", page);
        model.addAttribute("url", "/main");
        model.addAttribute("filter", filter);

        return "main";
    }

    @PostMapping("/main")
    public String add(
            @AuthenticationPrincipal User user,
            @Valid Message message,
            BindingResult bindingResult,
            Model model,
            @RequestParam(required = false, defaultValue =
"") String filter,
            @PageableDefault(sort = { "id" }, direction =
Sort.Direction.DESC) Pageable pageable,
```

```
            @RequestParam("file") MultipartFile file
    ) throws IOException {
        message.setAuthor(user);

        if (bindingResult.hasErrors()) {
            Map<String, String> errorsMap =
ControllerUtils.getErrors(bindingResult);

            model.mergeAttributes(errorsMap);
            model.addAttribute("message", message);
        } else {
            saveFile(message, file);

            model.addAttribute("message", null);

            messageRepo.save(message);
        }

        Page<MessageDto> page =
messageService.messageList(pageable, filter, user);
        model.addAttribute("page", page);

        return "main";
    }

    private void saveFile(@Valid Message message,
@RequestParam("file") MultipartFile file) throws IOException
{
        if (file != null &&
!file.getOriginalFilename().isEmpty()) {
            File uploadDir = new File(uploadPath);

            if (!uploadDir.exists()) {
                uploadDir.mkdir();
            }

            String uuidFile = UUID.randomUUID().toString();
            String resultFilename = uuidFile + "." +
file.getOriginalFilename();

            file.transferTo(new File(uploadPath + "/" +
resultFilename));
```

```
                message.setFilename(resultFilename);
            }
        }

        @GetMapping("/user-messages/{author}")
        public String userMessges(
                @AuthenticationPrincipal User currentUser,
                @PathVariable User author,
                Model model,
                @RequestParam(required = false) Message message,
                @PageableDefault(sort = { "id" }, direction =
Sort.Direction.DESC) Pageable pageable
        ) {
            Page<MessageDto> page =
messageService.messageListForUser(pageable, currentUser,
author);

            model.addAttribute("userChannel", author);
            model.addAttribute("subscriptionsCount",
author.getSubscriptions().size());
            model.addAttribute("subscribersCount",
author.getSubscribers().size());
            model.addAttribute("isSubscriber",
author.getSubscribers().contains(currentUser));
            model.addAttribute("isCurrentUser",
currentUser.equals(author));
            model.addAttribute("page", page);
            model.addAttribute("message", message);
            model.addAttribute("url", "/user-messages/" +
author.getId());

            return "userMessages";
        }

        @PostMapping("/user-messages/{user}")
        public String updateMessage(
                @AuthenticationPrincipal User currentUser,
                @PathVariable Long user,
                @RequestParam("id") Message message,
                @RequestParam("text") String text,
                @RequestParam("tag") String tag,
```

```
            @RequestParam("file") MultipartFile file
    ) throws IOException {
        if (message.getAuthor().equals(currentUser)) {
            if (!StringUtils.isEmpty(text)) {
                message.setText(text);
            }

            if (!StringUtils.isEmpty(tag)) {
                message.setTag(tag);
            }

            saveFile(message, file);

            messageRepo.save(message);
        }

        return "redirect:/user-messages/" + user;
    }

    @GetMapping("/messages/{message}/like")
    public String like(
            @AuthenticationPrincipal User currentUser,
            @PathVariable Message message,
            RedirectAttributes redirectAttributes,
            @RequestHeader(required = false) String referer
    ) {
        Set<User> likes = message.getLikes();

        if (likes.contains(currentUser)) {
            likes.remove(currentUser);
        } else {
            likes.add(currentUser);
        }

        UriComponents components =
UriComponentsBuilder.fromHttpUrl(referer).build();

        components.getQueryParams()
                .entrySet()
                .forEach(pair ->
redirectAttributes.addAttribute(pair.getKey(),
pair.getValue()));
```

```
            return "redirect:" + components.getPath();
        }
    }
```

```
package com.example.orange.service;

import com.example.orange.domain.Role;
import com.example.orange.domain.User;
import com.example.orange.repos.UserRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;
import org.springframework.util.StringUtils;

import java.util.*;
import java.util.stream.Collectors;

@Service
public class UserSevice implements UserDetailsService {
    @Autowired
    private UserRepo userRepo;

    @Autowired
    private MailSender mailSender;

    @Autowired
    private PasswordEncoder passwordEncoder;

    @Value("${hostname}")
    private String hostname;

    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        User user = userRepo.findByUsername(username);

        if (user == null) {
            throw new UsernameNotFoundException("User not found");
        }

        return user;
    }

    public boolean addUser(User user) {
        User userFromDb = userRepo.findByUsername(user.getUsername());

        if (userFromDb != null) {
            return false;
        }

        user.setActive(true);
```

```
        user.setRoles(Collections.singleton(Role.USER));
        user.setActivationCode(UUID.randomUUID().toString());
        user.setPassword(passwordEncoder.encode(user.getPassword()));

        userRepo.save(user);

        sendMessage(user);

        return true;
    }

    private void sendMessage(User user) {
        if (!StringUtils.isEmpty(user.getEmail())) {
            String message = String.format(
                    "Hello, %s! \n" +
                            "Welcome to Orange. Please, visit next link:
http://%s/activate/%s",
                    user.getUsername(),
                    hostname,
                    user.getActivationCode()
            );

            mailSender.send(user.getEmail(), "Activation code", message);
        }
    }

    public boolean activateUser(String code) {
        User user = userRepo.findByActivationCode(code);

        if (user == null) {
            return false;
        }

        user.setActivationCode(null);

        userRepo.save(user);

        return true;
    }

    public List<User> findAll() {
        return userRepo.findAll();
    }

    public void saveUser(User user, String username, Map<String, String> form) {
        user.setUsername(username);

        Set<String> roles = Arrays.stream(Role.values())
                .map(Role::name)
                .collect(Collectors.toSet());

        user.getRoles().clear();

        for (String key : form.keySet()) {
            if (roles.contains(key)) {
                user.getRoles().add(Role.valueOf(key));
            }
        }
```

```
            userRepo.save(user);
        }

        public void updateProfile(User user, String password, String email) {
            String userEmail = user.getEmail();

            boolean isEmailChanged = (email != null && !email.equals(userEmail)) ||
                    (userEmail != null && !userEmail.equals(email));

            if (isEmailChanged) {
                user.setEmail(email);

                if (!StringUtils.isEmpty(email)) {
                    user.setActivationCode(UUID.randomUUID().toString());
                }
            }

            if (!StringUtils.isEmpty(password)) {
                user.setPassword(password);
            }

            userRepo.save(user);

            if (isEmailChanged) {
                sendMessage(user);
            }
        }

        public void subscribe(User currentUser, User user) {
            user.getSubscribers().add(currentUser);

            userRepo.save(user);
        }

        public void unsubscribe(User currentUser, User user) {
            user.getSubscribers().remove(currentUser);

            userRepo.save(user);
        }
    }
```